University of Strathclyde
Department of Computer and Information Sciences

A New Heuristic-Based Model of Goal Recognition Without Libraries

by
David Thomas Pattison

A thesis presented in fulfilment of the requirements for the degree of

Doctor of Philosophy

June 10, 2015

Declaration

Signed:

Date:

License

SOYA MARGARINE

"Bread is bread."
"Just a minute. Let me think about that. Bread is bread. Is that what you said?"
"Yes. Bread is bread. I've been considering it for some time. Now that I'm sure, I've come straight out with it. My philosophy is in there somewhere."

"Hey, Lucy' "Yes, Dan?" "Bread is bread." "No kidding!"

Ivor Cutler
A Nice Wee Present From Scotland

Acknowledgements

I firstly must thank my supervisor Derek Long for his insight and support throughout my period of study. Both he and all members of CIS have been of great help in completing this work. Thanks also go to my new colleagues in EEE for similar support, and in showing me a world beyond plan recognition. To Ian Ruthven, I say thank you for the (substantial) support over the years — it is finally time to dust off those old stone records and chisel out "writing up" next to my name.

I must particularly thank Ron Petrick and Alex Coddington for their tireless work in evaluating this work. The term "rigorous and challenging examination" has never been more appropriately applied. Their job must be complete, as I never, ever want to go through the process again.

To all those fellow PhD students who I met along the way, I thank them for their discussion, support and encouragement, even though most of it was completely unrelated to any field of research. While many of us such as Pete, Tommy, Bram, Michael, Wasif and the Judge have gone our separate ways over the past few years, I hope that we remain in touch. If Vietnam veterans can manage to see each other every ten years, so can we — they had it easy in comparison.

Special thanks go to Alan, Alastair (Snr.) and Dave for their words of wisdom in dark times, and their help in reviewing this work. Their ability to be kind in the face of overwhelming mediocrity is unbounded.

To my mum, dad and sister, thank you for "not mentioning the war". Perhaps now you can finally approach strangers to tell them that your son/brother is a doctor — just not the useful kind. Maybe that can come next, now that I have some free time.

Finally, and most importantly, I have to thank my wife Adele for all her love and support over the past *cough* years. If it weren't for her asking why I'm sitting on the couch when I could be writing up, this process would never have ended. Of course, the true final thanks must go to my son Alastair. There is nothing like a screaming child to get one motivated to go elsewhere and write up.

For Alastair, who won the race (by a considerable margin).

Abstract

Goal Recognition concerns the problem of determining an agent's final goal, deduced from the plan they are currently executing (and subsequently being observed). For over twenty years, the *de facto* standard in plan and goal recognition has been to map an agent's observations to a set of known, valid and sound plans held within a *plan library.* In this time many novel techniques have been applied to the recognition problem, but almost all have relied on the presence of a library in some form or another.

The work presented in this thesis advances the state-of-the-art in goal recognition by removing the need for any plan or goal library. Such libraries are tedious to construct, incomplete if done by hand, and possibly contain erroneous or irrelevant entries when done by machine. This work presents a new formulation of the recognition problem based on *planning*, which removes the need for such a structure to be present. This greatly widens the scenarios in which goal recognition can be realistically performed.

While this new formalism overcomes many of the problems associated with traditional recognition research, it remains compatible with many of the concepts found in previous recognition work. This new definition is first defined in the context of a rational agent and observer, before several relaxations are introduced which enable tractable goal recognition. This relaxed implementation is then extensively evaluated with regard to multiple aspects of the recognition problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Recognition Problem

The ability of a system to perform accurate recognition upon the observations of an agent has long been a target of Artificial Intelligence (AI). Without an ability to comprehensively understand the true context of an agent's decision-making process, the system may be unable to react in the most appropriate manner.

Recognition can exist in many forms, from working with low-level sensor data through to high-level representations of the world. This can lead to a somewhat ambiguous definition of the term itself. It may simply mean an awareness of the environment in which the agent exists for use in its own decision making, or it can mean observing an external agent for a multitude of possible reasons, such as surveillance, assistance or competition. Furthermore, it may be performed in order to determine what the purpose of the plan observed *was*, or to predict what that purpose *is*. The output of these variants can then be used in many more applications which can further obscure the definition.

Traditionally, recognition can be split into several sub-categories which all seek to solve different (yet sometimes overlapping) problems. *Plan recognition* (PR) — which is often viewed as an umbrella term for all forms of recognition — is concerned with the detection of the *plan* which is being executed by an agent. If only the agent's final goal is of interest, the work is classed as *goal recognition* (GR), with the plan used to achieve this being somewhat irrelevant. *Intent recognition* is a pseudo-subset of PR and GR which aims to determine the agent's future actions or goals, although not necessarily the complete plan or final goal. Another view of this argues that intent recognition exists only when there is a dialogue between the observer and subject [21]. *Activity recognition* can be thought of as a specialisation of PR which uses human knowledge to abstract and classify certain aspects of the problem. For example, if the agent is seen entering a car, driving to the supermarket, parking then entering the supermarket, it may be reasonable to encompass these actions as part of a “shopping” activity. While plan and goal recognition can be encapsulated as low-level actions or facts, activity recognition places a more abstract meaning on the plan and/or goal. This ability to abstract and classify action sequences is a distinctly human quality, and one which machines struggle with, leading to activity recognition being heavily associated with fields such as robotics and assistive technologies [50, 68, 114].

This final point hints at an assumption often made in the recognition literature — the availability of prior, human-level knowledge. This is commonly encapsulated in a *plan-library* [97], a construct which contains a finite set of known, valid plans. Entries can explicitly specify plan features such as the goal, abstractions of actions, ordering requirements, disjunctions and probabilities.

1.1.1 Thesis Motivation

This thesis positions itself against the argument that a plan-library need be present for recognition to take place. Such structures are expensive to create in both time and resources required, limiting the ability to rapidly deploy a recognition-enabled system in the real-world.

To overcome the necessity of plan-libraries, this thesis describes a new and novel model of performing goal recognition by framing it as a *planning* problem. In doing so it tackles *pure* goal recognition. That is, there are no constraints or assumptions placed upon the agent being observed or the world in which they exist beyond those defined in the domain specification (action and fact templates, along with a grounded initial state). Moreover, there is no dependency upon a plan-library being available, although one can be implicitly accommodated if it is available. Thus, the model becomes applicable in scenarios where such structures are intractable or cannot be constructed.

1.2 Goal Recognition versus Plan Recognition

The ambiguity over which variant of recognition a piece of work falls into extends to the recognition literature itself. The most common debate is concerned with whether the problem being tackled is goal recognition or plan recognition. Work which is credited as being plan recognition rarely exceeds the accepted definition of goal recognition [60, 145, 146], yet the all-encompassing term of "plan recognition" remains a consistent presence. Given that the work presented is focussed on goal recognition, but also has aspects of plan recognition within the model, both terms must be clarified.

Concisely, goal recognition is the problem of simply determining *only* the goal of an agent. This goal may be the final goal within the plan being observed, or an intermediate goal which must be achieved prior to plan termination. Plan recognition can be seen to be a superset of GR, in which the plan being executed is also required. This plan can be used to determine the final goal of the plan, which should always be a subset or equal to all facts true in the state entered after the final observation is executed. Additionally, it is trivial to compute the intermediate states which the agent will pass through before the plan ends, and just as importantly, *when* the plan will end.

The model presented in the *implementation* of this work encapsulates a subset of pure PR, as shown in Figure 1.1. While the entire GR problem is included as part of the implementation, certain features of plan recognition which are conceivably part of the formal model are not explicitly implemented. For example, while it is possible to infer an agent's future actions, these are considered discrete and may not necessarily form a causally-linked plan as would be the case in pure PR.

A further clarification is required when discussing *libraries*. In the context of this work, a "library" shall be classed as "a finite repository of domain-specific information which is available prior to and/or during the recognition process, and further, is represented explicitly within memory". In the case of a plan-library, the information contained would be at least a set of plans and goals. For goal recognition, only goals would be stored. This leads to a distinction between the three terms. *Plan libraries* are a superset of *goal libraries*, and a *library* is simply anything which meets the definition above. In either case, the library can have been constructed by hand [60, 70, 97] or automatically by machine before recognition begins [18] or at runtime [19, 112], and assigned prior probabilities in a similar manner.

The potential to automatically generate plans for libraries or agents in general, hints at a previously only loosely explored avenue of research, wherein plan recognition and planning are *integrated*.

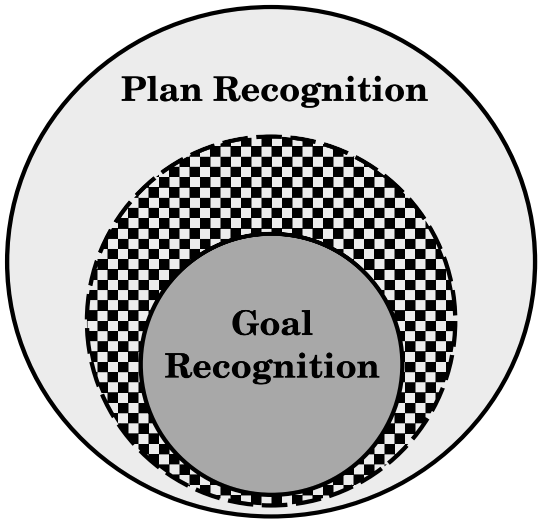


Figure 1.1: A visual representation of the model presented in this thesis, in the context of plan and goal recognition. The checked area indicates that the features presented are a superset of goal recognition, but a subset of plan recognition.

1.3 Planning and Recognition

While recognition is the process of extracting a hypothesis with regard to the agent's observed behaviour, *planning* is the process of constructing the plan itself. Like recognition, this is also an active area of AI research, and many parallels can be drawn between the two.

Viewed abstractly, both planning and recognition problems share the same world representation. For a given problem, both exist in the *state-space*, $\$$, in which the application of *actions* moves the agent from a predecessor state, S, to a successor state S'. The task at hand for planning is to find the shortest path through this space to a state S'' which contains the agent's goal $G \subseteq S''$. Recognition strives to determine which of the states containing the agent's goal, $\$_{goal}$, they are trying to reach $G \subseteq S'' \in \$_{goal} \subseteq \$$.

In *complete* plan recognition where an agent is known to be optimal and a plan-library of optimal plans is available, members of this library which are inconsistent with the observed actions are eliminated until a minimal set is left. This set must contain the true plan and goal, or the library is incomplete. Goal recognition in $\$$ using a plan/goal library is also a process of elimination, in which states containing goals that could have been achieved prior to the current timestep are eliminated.

The parallels between planning and recognition may not be apparent at first, but consider that in both of these cases an agent is simply transitioning through the state-space. In both cases, *it is reasonable to assume that they will do so in the most optimal manner.* It is this assumption which will drive many of the core arguments of this thesis.

1.3.1 Heuristic Guidance Through a State-Space

For planning, optimality is determined by which action is chosen at each decision point (successor state), with this being heavily or completely determined by the agent's *heuristic*. In an optimal or *rational* agent, the heuristic value computed in each state will always be perfect and can be used to derive optimal plans to any goal.

In recognition, the reverse of this is true. The observer is trying to deduce the agent's goal by determining which plans or goals *best fit* the agent's observed behaviour thus far. That is, the observer can apply their own heuristic to the agent's behaviour, in order to determine which goals are having their heuristic estimate lowered by the actions of the agent.

As a simple visual example, Figure 1.2 shows a small grid world state-space in which states are equivalent to goals ($S_i = G_i$). The agent starts at the top-left corner and wishes to move to the bottom-right state (which contains its goal, $G = S''$). In the role of the agent, there are three possible successor states at each decision point — one to the right, another to bottom-right and one below. The agent uses its heuristic to determine that the bottom-right state will always provide the lowest estimate of work remaining to the goal, and therefore the action which achieves it is always selected.

Now consider the case of the observer. At problem initialisation, they can see that all states are potential goals as no behaviour has been observed yet[1]. After the first observation, they can rule out those states which were to the immediate South and East of the initial state, as their heuristic value after the observation has not lowered. After three observations the observer can see that only a few states remain which can be reached in an optimal fashion from the initial state (Figure 1.2b), therefore this set of states must contain the agent's goal. Finally, by the time Figure 1.2c is reached, there are no more states remaining which could have been reached optimally, so the goal must be within this state.

By modelling the recognition process as one of heuristic estimation, there

[1]The potential for the agent *already* being in a goal state at problem initialisation is discussed later in this work.

is no need to construct a plan-library. Goals and states can be enumerated at runtime, given a well-formed problem input. The primary task then becomes one of substituting a suboptimal heuristic in place of the optimal one, as these are NP-hard to compute [32].

In the same way that the application of suboptimal heuristics in planning has allowed for rapid expansion of the field and served to increase its wider applicability, this thesis states that a similar procedure to that described above can also reap the rewards of modern domain-independent heuristics when used in goal recognition.

1.4 Thesis Contributions

The work presented in this thesis advances the state-of-the-art in several areas which have been overlooked by prior work in GR or modelled in such a way as to restrict their application or formal evaluation.

The model presented is the first that has the ability to perform goal recognition with no prior assumptions of the domain, plans or agent being observed. While other models can theoretically do this, they require a library of goals which makes real-world application intractable. By representing the GR problem as one of planning, *any* combination of facts can be considered as a goal and presented as a hypothesis. This allows the entire *goal-space* to be represented using fewer facts, which in turn reduces problem complexity and removes the requirement of a plan or goal library being constructed prior to recognition beginning. This culminates in the model being able to perform recognition on any domain, without prior knowledge of its existence.

Another contribution of this work is that the agent being observed need not be *rational* — that their plans will always be optimal. In fact, as prior information is not available, knowledge of how long the agent's plan will be at any time must therefore be unknown. However, if the length of the agent's plan *is* known, or at least the number of remaining observations, the goal-space can be analysed to detect which facts are achievable within the remaining $n \geq 0$ plan steps. Therefore, the number of remaining observations is estimated based on what the current goal hypothesis is. Whilst allowing for more accurate hypotheses, this further allows the prediction of *intermediate* states which represent the most probable state within these n timesteps. All of these features are novel aspects of the model which further the state-of-the-art of both the goal and plan recognition literature.

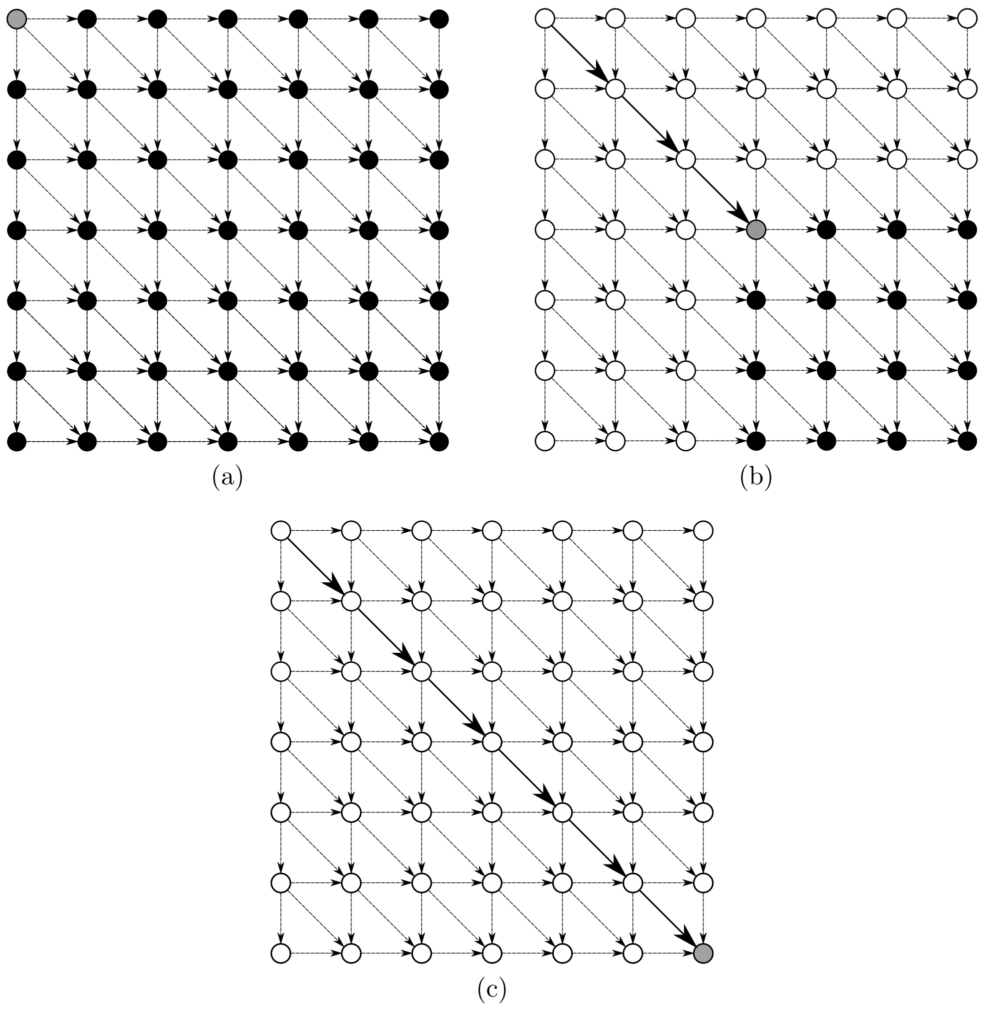


Figure 1.2: An example of an agent transitioning through a state-space from the top-left corner in Figure 1.2a to the bottom-right in Figure 1.2c. States which are still reachable by an optimal plan are shaded black, while those which this no longer applies to are shaded white. The current state in each figure is shaded grey.

A consequence of suboptimal agents is the possibility for goal *abandonment*. Here, the agent executes part of the plan which achieves their original goal, before transitioning to another plan which achieves a different goal. Here, the abandonment of the original goal can be viewed as an extreme case of suboptimality in the plan. Previous work on abandonment has used a plan-library as a means of determining which actions are expected to be observed, given that a set of candidate plans are being executed [61]. This naturally suffers from the problems outlined above, with the ability to detect goal abandonment linked directly to the completeness of the library. Therefore, this thesis presents a means of detecting abandoned goals as a side-effect of the observation process, something which has not been considered before.

Previous probabilistic recognition models based on libraries have had the benefit of being able to assign *prior* probabilities to each plan/goal before recognition begins [18, 38, 70]. This is often done by a domain-expert or by training the underlying system using example problems and plans, both of which are time consuming tasks. As the model presented has no such library, it therefore must automatically generate prior probabilities using *domain analysis* — another novel contribution of this work. While domain analysis has become common in the planning community [51, 78, 88, 138, 148], run-time analysis of the state-space is virtually unknown in the recognition community. As the following chapters will show, domain analysis becomes possible only once plan-libraries are removed and the underlying problem structure examined.

Finally, throughout all previous recognition work *standardisation* of input, output, testing and scoring metrics has been largely unique to the author and model presented. This in turn leads to ambiguity in the evaluation and comparison of different models and implementations, which further adversely effects the perception of recognition literature. This work attempts to try and partially tackle the former of these problems, by first accepting problem input in a well-known and standardised format [55, 120], and evaluating the model using the most widely applied techniques.

In summary, the contributions of this thesis are as follows.

- To enable goal recognition without the availability of a plan-library.
- Allow suboptimal agents and plans to be observed, and further to allow the possibility of the agent abandoning goals.

- Automatically derive prior probabilities for goals without knowledge of the domain.
- Accept any problem which adheres to a formalised input specification.

1.5 Statement of Intent

The core concept of the model proposed herein is that it should be possible to determine an agent's goal purely by observing their movement through the state-space in which they exist, primarily using heuristic estimation. This removes the need for any assumptions to be constructed about the agent, and casts the problem as the purest form of *keyhole recognition* [1, 70], in which the agent has no knowledge of being observed or interaction with the observer.

This is achieved by applying techniques normally associated with planning, wherein a heuristic is used to guide the search process towards the goal, in order that the minimum series of actions are executed. In this work, the planning model is inverted, such that the heuristic is used to determine which areas of the state-space are being moved towards. This is used as an indicator that the agent's true goal lies within this region, which in turn can be used to construct a *hypothesis* of this final goal and intermediate goals which must be achieved prior to this.

Beyond this primary functionality, a new method of extracting the set of most probable goals prior to observation beginning, using only domain analysis is presented. This is again achieved using existing planning techniques.

1.6 Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 motivates the argument for a move away from the traditional library-based approach to recognition, to one which can perform recognition without the need for such a structure. A description of the most widely accepted models in both plan and goal recognition is provided, along with an introduction to planning using heuristic search. The core milestones of this work are then set out along with the rationale behind their choice.

Chapter 3 formally defines the new model of goal recognition by first noting the similarity between planning and recognition, such that a common formalism can be constructed. This is then used to present a *heuristic-based* goal recognition model.

While Chapter 3 formalises the new model, Chapter 4 provides details of its implementation — IGRAPH. This is only possible by applying several *relaxations* to the model, which enable tractable goal recognition to become possible. Techniques derived from the relaxed implementation are then used to automatically construct a non-uniform *initial probability distribution.*

Evaluation of various components implemented in IGRAPH is reported in Chapter 5. This focuses primarily on the accuracy of *intermediate* and *final* hypotheses, which are generated after each observed action. The impact of the *domain analysis* carried out prior to recognition beginning is also reported.

Finally, Chapter 6 concludes the thesis by returning to evaluate the system against the requirements outlined in Chapter 3. Potential for further expansion and exploitation of the model is also provided.

1.7 Related Publications

- David Pattison and D. Long. Accurately determining intermediate and terminal plan states using Bayesian goal recognition. In David Pattison, Derek Long, and Christopher W. Geib, editors, *Proceedings of the First Workshop on Goal, Activity and Plan Recognition (GAPRec)*, pages 32 – 37, June 2011

- David Pattison and Derek Long. Extracting plans from plans. In S. Fratini, A. Gerevini, D. Long, and A. Saetti, editors, *Proceedings of the 28th Workshop of the UK Special Interest Group on Planning and Scheduling, PLANSIG'10*, pages 149 – 156, December 2010

- David Pattison and Derek Long. Domain independent goal recognition. In *STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium*, volume 222, pages 238 – 250. IOS Press, August 2010

Chapter 2

Background and Motivation

2.1 Overview

This chapter provides a description of the motivation which drives the work presented in this thesis. This is performed by providing an overview of the existing literature on plan and goal recognition, segmented according to the major components of the recognition problem and how they are tackled.

The chapter begins with a brief historical overview of automated planning, as this will form the basis of many common principles used throughout this thesis. This is then followed by an overview of historical and contemporary work in plan and goal recognition. Both sections detail the major models of each respective field and highlight any overlap in common representations.

Historical work is used as the motivator for a new model of recognition in Section 2.4. This highlights the various aspects of the recognition problem which current models only partially achieve or ignore. This forms the basis of the model presented in Chapter 3.

2.2 Planning

The work presented herein is heavily based on planning methodologies applied to recognition. Therefore, it is prudent to provide a brief overview of this field and specific areas which relate to this thesis.

As the introductory chapter stated, planning can be viewed as the mirror of plan recognition. While recognition aims to determine the plan or goal of an agent by observing a plan, planning is the generation of this plan in an efficient manner. In the case of PR with a plan library, the mirrored planning process would be akin to a focussed heuristic search through only portions of the possible state-space (i.e. those plans existing in the library).

Planning exists in many forms including simple propositional planning [23, 27, 79, 88]; metric and temporal planning [42, 83]; probabilistic [22, 25, 163]; conformant and contingent planning [87, 128]. Each of these fields can be generalised to the creation of a plan given a set of hard or soft constraints. These may take the form of goal literals; limited numeric resources or temporal orderings amongst others. While a constraint is intended to affect the plan created in some manner, this is implicitly assumed to be the lowest cost solution.

2.2.1 Classical Planning

In classical planning the world is represented using propositional logic[1], as popularised by the STRIPS formalism [49]. No metric resources or temporal aspects are present, and the world is fully-observable and deterministic. Each action has an associated cost, which is usually assumed to be uniform across all actions.

The purpose of planning is to generate a minimal set of ordered actions which transform the initial state I into a state S' containing all specified goals $G^* \subseteq S'$, although in practice any ordering of actions which achieve G^* is acceptable. This positions planning as a directed-graph search problem, where nodes equate to states and each edge $a^i = \langle S, S' \rangle$ equates to an action which if applied in S transforms the state to S'. Therefore, the problem is how to guide the search through this state-space $\mathbb{S}$, as enumeration for non trivial problems is not tractable due to the *curse of dimensionality* [32].

This problem of state-space explosion is most commonly tackled by using a *heuristic* to guide search towards the goal with minimal enumeration of $\mathbb{S}$. If the heuristic is *domain-dependent* then the planner is similarly classed and will likely perform well only on specific problems [7, 106, 126, 127]. However, if the heuristic is *domain-independent* it can potentially perform well on a variety of unseen domains. It is this latter class which will be used in this work.

[1]Note that other forms of planning exist, such as using *constraint satisfaction problems* (CSP) or *satisfiability* (SAT) problems [46, 156]. While these can be translated into a STRIPS-esque form and vice-versa, this thesis will concentrate solely on graph-based approaches. As there is a one-to-one mapping possible between all of these models, it is conceivable that the work presented can be implemented in such forms.

2.2.2 Domain-Independent Heuristics

The arrival of informative domain-independent heuristics in planning [27, 28, 119] enabled a variety of problems to be solved using a single heuristic within a planner. These heuristics use the structure of the state-space to compute an estimate of how much work remains to achieve a specified goal. In the case of classical planning, this estimate is simply the number of actions required to achieve the goal from a specific state.

An informative heuristic can result in a massive reduction in search time and number of successor states expanded whilst searching for the goal, versus a brute-force expansion of all states until the goal is found [23]. However, in this form the problem is intractable due to its complexity [32]. Thus, the problem is often *relaxed* by ignoring certain components such as the delete-effects of actions [28, 88]. Regardless of the technique used, domain independent heuristics[2] are based upon computing an estimate $h(G^*)$, of the work required to achieve a goal G^*, based upon the available action set.

In addition to simple heuristic estimates, some additional problem-solving abilities may be implemented to aid in minimising search. For instance, Hoffmann's work on the FAST FORWARD (FF) planner and associated heuristic [88] incorporates the concept of *helpful actions* to guide search. These are actions which are applicable in the current state and achieve at least one of the goal literals found in first layer of the *relaxed plan*. Given two successor states with equal heuristic estimates, if one is achieved using a helpful action, it will always be selected as the best option. This domain-independent functionality contributes strongly to the performance of FF.

2.2.3 Domain Analysis

Early work on planning used *only* the heuristic estimate during the search process [23, 28]. While this is an acceptable approach if the heuristic is known to be optimal, in practice additional information is often used to guide search. This is often done through incorporating modest amounts of *domain analysis*. That is, prior to search starting, the domain and problem structure are analysed for any useful information which can speed up or optimise planning. For example, approximation of domain invariants enables faster planning through lowering the number of reachable states considered in search [16, 51, 53, 64, 65, 149], while

[2]From this point onwards, the term "heuristic" can be assumed to refer to the domain-independent variety, unless otherwise stated.

detection of goal-ordering and landmarks (literals which must be true in every plan for a given goal) often allows the number of expanded states to be minimised [103, 115, 148].

Indeed, the majority of modern planning systems use some kind of domain analysis to make the search process more focussed [54, 115, 116, 138, 148]. While potentially expensive, this can result in a performance increase on certain classes of problem, and enable solutions to be derived on other wise intractable problems.

Domain analysis in planning often aims to make difficult problems tractable by guiding search [148, 161] and ignoring areas of the state-space which are irrelevant. This principle can also be applied to recognition. Given an unseen problem, the structure of the domain will make certain goals more probable than others, giving the recogniser the potential to produce better hypotheses prior to and during observation. This is in contrast with existing work in recognition, which does not use domain analysis.

2.3 Historical Background

2.3.1 Plan Recognition

Modern plan and goal recognition can trace its origins back to the study of human psychology and behaviour in problem solving, largely explored through understanding in natural language parsing scenarios [12, 34, 48, 135].

The first formal definition of this was by Schmidt *et al.* in 1978 [150], who used natural language processing as an application of recognition. Their formalism dictated that a strict language is required for performing recognition tasks in order for all experiments to be sound. Their work also stated that it is necessary for the recognition system to reflect the assumptions of the user/observer, such as the information available to and used by the subject; the subject's knowledge at problem instantiation; and a model of how the world transforms as a result of the subject's actions, or more generally, simply how the world behaves.

The influence of human psychology on the work of Schmidt *et al.* is evident in their further assumptions. For example, it is assumed that the observer will only carry a single hypothesis at any time, and that it is often more likely that this will be *generalised* based on the observed evidence not being specific enough — it is better to say that the subject is getting *something* out of the fridge, rather than being forced to choose amongst all items in the fridge. This *lifted* hypothesis ("get *something* from the fridge") can then be refined after more evidence has been provided, to produce a *grounded* hypothesis (“get *milk* out of the fridge”).

The work of Schmidt *et al.* and other early PR pioneers [135] has been extended throughout the intervening years. This has led to a divergence in many of the current models of PR, which seek to target differing areas of the PR problem. However, at a more abstract level, these models fall into one of three categories: *keyhole*, *intended* and *adversarial* recognition.

In *keyhole* recognition [41], the agent is assumed to be unaware of the observing agent, and cannot interact with them in any way and lends itself towards applications such as security surveillance [5] or deriving user goals in a game-environment [1, 74, 75]. Unlike this, *intended* recognition [41], allows the subject and observer to interact — specifically, the recogniser is allowed to query the subject. For instance, the recogniser may ask for the rationale behind a subject’s actions, or for assistance in deriving the current world state. Such work is often related to natural dialogue systems [2, 91, 122] or situations in which a system is designed to assist the user [13].

Finally, in *adversarial* recognition [59], it must be assumed that the agent is aware of the observer, and that they may attempt to explicitly interfere with the recognition process. This can be done through attempting to “cover up” actions executed or by intentionally leading the observer to believe that a plan/goal other than the true one is being pursued. While *keyhole* recognition is acceptable if the subject does not know they are under surveillance (such as CCTV monitoring), *adversarial* recognition is a more likely model to adopt in some scenarios — such as detecting terrorist behaviour [94], network intrusion monitoring [29, 60, 142] or video/serious-game opponents [95, 104].

Together, these three areas encapsulate all work in plan recognition. However, the underlying models used can differ greatly, as each attempts to target different areas of the PR problem. The following section provides an overview of the most widely accepted models, grouped according to their most prominent features (although these regularly overlap one another).

Logic-Based

Early work which followed on from Schmidt [150]; Perrault [135] and Allen *et al.* [2], naturally derived ideas from these works. This previous work focussed on the assumption that if plan-recognition is perfect in both its observation of the agent and in any conclusions drawn, it becomes a process of eliminating plans which are not *consistent* with the observations.

The work of Kautz and Allen in the 1980s, which is widely viewed as the first formal model of plan recognition as it is viewed today [97, 98], adopts this approach via various assumptions about the world and agent. This model frames plan recognition as a process of maintaining a consistent set of entries in a *plan-library* (via circumscription), based upon observations of the subject and a set of rules associated with the problem. A plan-library contains a set of *hierarchical task networks*[3] (HTN) [125], which represent the decomposition of a single, high-level goal into subtasks and atomic actions. In Kautz and Allen's work, this collection must be *complete*, meaning it must contain every possible plan for a given problem — or rather that it must contain every plan which is regarded as a valid output for the problem.

This final point is made possible by the assumption that *all* actions are executed for a reason. That is, that the subject demonstrates *rationality* or *bounded rationality*. In the former case, the agent will always execute an optimal plan, while in the latter they may not perform the optimal plan to achieve their goal as defined by an external observer — but instead will execute the plan to the best of their ability and knowledge.

Kautz and Allen's model overcomes many of the problems associated with prior work such as that of Schmidt *et al.*, by allowing for multiple, concurrent plans, which may share actions, to be considered. However, the major contribution of this work is arguably the concept and use of a plan-library in performing recognition. This allows the problem to become one of eliminating *inconsistent* plans given the set of observations so far. A plan is inconsistent if an observed action cannot be mapped to it given previous observations. The assumption of a plan-library or similar structure[4] being available to the recogniser has become the *de facto* standard approach to recognition [35] and has since framed the vast majority of both plan and goal recognition literature [6, 18, 30, 37, 38, 60, 70, 112, 141].

Unfortunately, the richness of Kautz' model means that the runtime is exponential in the size of the plan-library. This results in only trivial problems being solvable. The logic-based approach to the model also means that if multiple plans are returned as valid candidates, they are considered as equivalent, despite some potentially being highly unlikely.

[3]There are perhaps some minor differences between Kautz and Allen's model of an HTN and the modern definition, but these are trivial and for ease of presentation can be overlooked.

[4]Here, a "similar structure" to a plan-library is a data structure which contains a set of previously known plans and/or goals. The underlying representation (such as HTNs), is irrelevant.

Probabilistic and Bayesian

While the logic-based and grammatical model advances the recognition problem, it suffers from many restricting assumptions which limit the scope of potential application. For example, all consistent plans are equally likely despite some potentially being highly improbable. The output of the recognition process is also assumed to be a single hypothesis, which prohibits the potential for multiple explanations for the subject's actions, and also prevents multiple goals and plans being considered[5].

Given these drawbacks, it is therefore natural that a move towards *probabilistic* recognition evolved in subsequent years. This was first argued by Charniak and Goldman [37, 38] (C&G), who stated that without probabilistic models, ambiguity between hypotheses was inevitable and that it is better to output multiple probable hypotheses rather than a single hypothesis. However, the core of their argument is that the recognition problem is one of dealing with *uncertainty* — both in the world and agent observed.

The immediate effect of this is that plans can have an associated prior probability, indicating the likelihood of them being the true plan before observation has begun. For instance, if the only observation known is the subject entering a shop, then it is more probable that they are performing the "shopping" plan, rather than the "rob store" plan. However, C&G also allow for multiple hypotheses to be presented on the rationale that it is better to present several dissimilar hypotheses than a single unlikely hypothesis.

C&G's model was one of the first to incorporate Bayesian logic in performing PR reasoning — something which has become commonplace in later models and applications [1, 3, 31, 139, 140, 152]. Subsequently, the PR community has largely adopted Bayesian logic as standard, although *frequentist* approaches have also been explored [14, 33].

In particular, graphical models [124, 133, 143] have been heavily applied as a means of inferring agent intent [31, 74, 105]. C&G themselves automatically construct a Bayesian network for application in a story-telling environment [37, 38, 69]. Here, high-level goals are represented as root variables in the graph, with propositions derived from the story linked to these via a causal arc. As with much Bayesian research, a large volume of evidence is required to derive the conditional

[5]While Kautz and Allen's model supports concurrent plans and actions, these must be contained within a single HTN. It is not possible to consider multiple goals and associated plans across multiple HTNs.

and prior probabilities.

One of the most commonly cited model of recognition was formalised in 1999 by Goldman, Geib and Miller (GG&M) in their PHATT system, which is in fact a combination of a logic and probabilistic-based model [70]. Motivated by a wish to assist human subjects, it differs from the previously discussed models by taking the *state* of the agent into consideration. This builds upon the previously discussed models by allowing hierarchical plans to include partial-orderings between actions, and *partial observability.* As actions are observed, a *pending set* is constructed which contains all actions which could potentially follow the previous observation. These are extracted from members of the plan-library which are consistent with the observations thus far. If an action is observed which is *not* a member of the pending set, this can be used as evidence of a previous observation being unobserved. The pending set can then be used as a source of actions which the system can itself potentially execute, without breaking the observation/inference process.

In addition to this, GG&M's model includes some potential for the agent behaving *irrationally.* That is, the agent may perform actions simply "for the sake of it", where these actions have no effect on achieving *any* goal in the plan-library.

GG&M's work is implemented through an *abductive* reasoning system [136], which essentially models *Occam's razor* — the principle that the simplest explanation is the most probable. In the case of PR, this is naturally the plan which best maps to the observed actions with the fewest assumptions. In GG&M's implementation, this is done through a hand-coded rule-set, where each rule encapsulates a feature of the model (negative evidence, pending set etc.).

If left at this point, the model would potentially be *consistency-based* (if the plan observed were optimal). However, GG&M further incorporate probabilities into the model by associating a real value against the probability that the agent will perform some action for its own sake; that the agent will choose a particular sub-plan for a given goal; and that the agent will choose a particular action from a pending set of actions. In doing so, GG&M encapsulate the functionality associated with C&G's prior model.

Grammar-Parsing

Following on from Kautz and Allen's formal definition of PR, Vilain constructed a correspondence between this model and one of grammar parsing [158, 159]. The relationship here is that if a plan is constructed of several actions, it becomes akin to deriving meaning from a sentence in some formalised grammar, where each action is a lexical token and the plan-library defines the grammar. Partial plans can subsequently be represented by shorter sentences and atomic actions by terminals in the target language.

Context-free grammars (CFGs) [90] are particularly suited through application of *production rules* which form the basis of the language. Here, the CFG is defined as a tuple, $G = (\Sigma, N, S, P)$, where Σ and N are finite, disjoint sets of terminal and non-terminal symbols respectively, $S \in N$ is the start symbol and P is a set of production rules. Each rule takes the form $E \rightarrow \Lambda$, where $E \in N$ indicates that E must be true in order for tokens $\Lambda \in (\Sigma \cup N)^+$ to be achieved.

Given this definition, the transformation of a library-based plan recognition problem into a CFG formalism is obvious, with actions replaced by rules, goals by terminal/non-terminals and the initial state replaced by the start symbol [62].

By encoding the problem using *context-free grammars* (CFG) and using relevant newly-applicable algorithms, Vilain shows that a subset of Kautz' model can become solvable in polynomial time. However, to achieve this speed increase all plans must now be totally-ordered and cannot share/interleave actions.

Like Kautz and Allen's model, all hypotheses are equally probable, which leads to the previously discussed drawbacks. Pynadath and Wellman [140, 141] therefore use *probabilistic* context-free grammars (PCFGs) [36, 72] and later probabilistic *state-dependent grammars*, where production rules now have an associated probability $\mathrm{P}(E \rightarrow \Lambda)$ indicating how likely rule E is to be expanded into string Λ. This is not used in conjunction with typical PCFG algorithms, but rather to construct a dynamic Bayesian network [124, 133]. These are used to derive a probability distribution over the range of possible plans currently being executed, but in doing so are limited in their treatment of plans containing loops.

Further models have also been derived by Geib, Maraist and Goldman [63], and later by Geib alone [56, 57, 58] which extend the relationship between PR and natural language processing (NLP). The former takes GG&M's earlier PHATT model and optimises it by pruning matching plan-trees, sometimes reducing runtimes by an order of magnitude. The latter targets the follow-on problem of the *pending set* in GG&M's work potentially growing to an exponential size through committing to a hypothesis early in the observation process. This is done by translating the problem into a *combinatorial categorial grammar* [151], and pre-selecting critical actions from each plan[6] that can be used to minimise the pending set by rejecting candidates whose critical action has not yet been observed. This minimisation allows for extremely fast runtimes when the critical action is located on the rightmost token/observation, although in doing this the system loses predictive capability as the true hypothesis has its commitment delayed until the final observation.

GG&M's model is perhaps the most complete for library-based plan recognition published thus far. It incorporates many facets of prior models (such as plan interleaving and probabilistic reasoning), alongside novel features aimed at solving previously ignored or overlooked problems (negative evidence, agent state, plan intervention). However, it is not without flaws. There is no concept of unachievable actions (i.e. dead-ends or states which are no longer reachable due to previous observations), and the assumption of a plan-library remains, which now also requires a set of pre-assigned probabilities. GG&M also do not provide an implementation of the theory, but rather use a probabilistic horn abduction theorem prover [136] to validate the various rules outlined in the paper [70].

2.3.2 Goal Recognition

Until recently, goal recognition was often considered only in the wider context of plan recognition. Plan recognition systems which rely on a plan-library often have the goal for each plan explicitly specified [1, 6, 20, 37, 60, 70, 97], making selection of a goal hypothesis a side-effect of selecting a plan hypothesis. A major advantage of this model is that the number of goals can be relatively small when compared against the number of plans that are capable of achieving them. For example, *completeness* in Kautz and Allen's formalism refers only to the set of plans for each goal. Without this assumption, the complete set of goals for a problem is equal to every valid combination of literals which are reachable from the initial state.

While plan recognition makes use of a plan-library, a similar (or identical) structure is often used in goal recognition to limit the number of goals considered. Like plan recognition, the construction of this structure is often done by hand, but as there is no *ordering* to the data (each goal is often mutually-exclusive), automatic generation becomes a tractable possibility [19, 112].

Like plan recognition, goal recognition can be classed as *keyhole*, *intended* or *adversarial*. As this work concentrates on keyhole recognition, associated work

[6]Termed *plan heads* in [58], which are analogous to *head words* in an NLP sentence.

shall frame much of the following discussion on prior research. However, a further distinction is also possible. Blaylock [18] states that a goal recogniser can also be *flat* or *hierarchical*. In the former case, this means it can recognise only the top-level goal of the plan, while in the latter ordered (or partially-ordered) *sub-goals* of the overall goal can also be recognised (similar to a HTN).

As work in GR has progressed alongside PR until recently, it is simpler to group prior research by major authors in the field, rather than by evolution as in the case of PR above. Therefore, the following section instead lists the work of the primary researchers in GR, in approximately chronological order. The smaller nature of the GR research field also allows for further detail on prior work to be provided.

Lesh and Etzioni

Early work by Lesh and Etzioni [111] modelled goal recognition in a similar manner to Kautz and Allen's definition of plan recognition. That is, goals are eliminated based on their *consistency* with the actions observed. This is derived from a fully-connected *consistency graph* — a graph in which nodes represent all action and goal schemas, along with any observed actions, and edges indicate a dependency between nodes. Here, the term "schema" is analogous to a *lifted* action representation in planning or, alternatively, a collection of all possible grounded instances of an action. Observations are equivalent to a single *grounded* action.

After each observation the consistency graph is updated to eliminate goals and action schema that are no longer connected to the main graph (therefore being classed as *inconsistent*). Edges are removed based on a set of rules, such as eliminating the link between observed action a^i and unobserved action a^j if $(a^i_{add} \cup a^i_{del}) \cup a^j_{pre} \neq \emptyset$, where a^i_{add}, a^i_{del} and a^j_{pre} are the respective action add effects, delete effects and preconditions. That is, there is no causal link between them. Goal and action schemas which are no longer connected to the main graph are then pruned from consideration and future hypotheses.

Lesh and Etzioni have also explored adapting recognition to a previously unseen agent's plan with the *ADAPT* system [109]. The recogniser is trained using an agent's recent behaviour which has *not* been annotated with the agent's true goal. This data is then used to try and find the combination which provides the best results. Unlike the previous system, this work does not assume access to training data or require a policy to be constructed for recognition. However, it is noted that the *ADAPT* system is intended to be an enhancement to an existing system and not a full goal-recogniser.

Outwith this prior work, Lesh and Etzioni were amongst the first to tackle automatic generation of plan libraries for application in goal recognition [112], with the work focussed on the problem of scalability in plan and goal recognition. They use automatic generation of *complete*[7] plan libraries and represent these plans in a compact fashion in order to reduce the space requirements. Each goal an agent has is mapped to a single plan in the library, with agents assumed to be executing between 1 and n plans in order to achieve n goals.

Blaylock and Allen

The problem of automatically obtaining plan libraries/corpus[8] has also been tackled by Blaylock and Allen [19]. This is accomplished by using the SHOP2 planner [127], modified such as to stochastically generate hierarchical plans [125] (see Figure 2.1a) for a random (but valid) initial state and goal. The type of goal generated for planning is derived by hand-coded probabilities which are provided *a-priori*. While no results are given, the authors do note that the quality of the resulting library is highly dependent upon the quality of the planner and algorithms used to seed the initial state and goal. In particular, deriving a library using a planner may not have a strong correlation with human behaviour, which much PR work is targeted at.

Aside from this work, Blaylock and Allen have explored other areas of the goal recognition problem. Continuing the strong relationship between recognition and language, they have used NLP techniques in the context of goal recognition [18], by using *n-gram* models along with a plan corpus to estimate the true goal. Here, the prior probability of a goal being the true goal is determined by analysing the plan corpus before recognition begins, in the same manner as the probability of a word appearing in a text document can be determined in NLP. To compute these probabilities they apply both *unigram* and *bigram* models, which state that in the former case, an observation is only dependent upon the goal, and in the latter that it is further dependent upon the previous observation[9].

[7] *Complete* in this context rules out any plans and goals which will never be attempted or achieved.

[8] Blaylock and Allen often refer to the plan or goal library as plan or goal *corpus*. In this thesis, these terms can be viewed as interchangeable.

[9] To accommodate the situation where only a single action has been observed, the plan is always annotated with a "start" action stub. This allows information on which true actions tend to be applied first in a plan to be obtained.

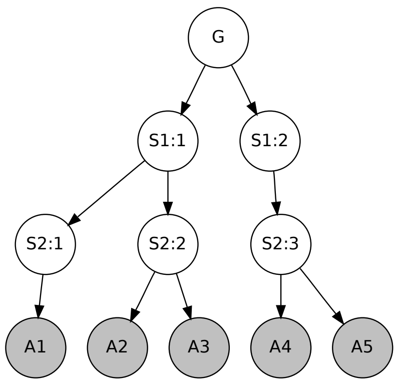
In practice, the plan corpus used is Lesh's Unix plan-library [109]. This contains 11 unique goals and 412 possible actions, which have 22 underlying action types. The recogniser was able to successfully identify the final goal with 55.4% and 55.6% accuracy for the unigram and bigram models respectively. However, the average plan length is only 7 steps long, with results for larger domains not given. In addition, while the system is aimed at rapid deployment, the problem of data collection is highlighted by the authors as a potential barrier to this. This observation led to the work on automatic library generation discussed above [19].

Later work by the authors has applied machine learning techniques to goal recognition [20], in order that recognition is no longer considered to be inferring only the agent's final goal, but also any intermediate goals. The SHOP2 planner [127] is again used as the source of hierarchical plans, but these are no longer considered in their original form. Instead, each plan is decomposed into a set of ordered sub-goals which are equivalent to a depth-first traversal of the original plan, starting from the root node as shown in Figure 2.1b. Each of these *goal chains* is used to form a cascading hidden Markov model (CHMM) — a structure related to hierarchical hidden Markov models (HMM) [123], in which HMMs [143] are stacked upon one another. The emissions of hidden state i in layer D are equal to the observed action at timestep i. These hidden states are then used as the emissions of layer $D - 1$, a process which repeats until layer 1 is reached. The depth of the CHMM is equal to the size of the longest plan of any HTN in the corpus, with any variable-length plans being padded out to match this maximum depth [21].

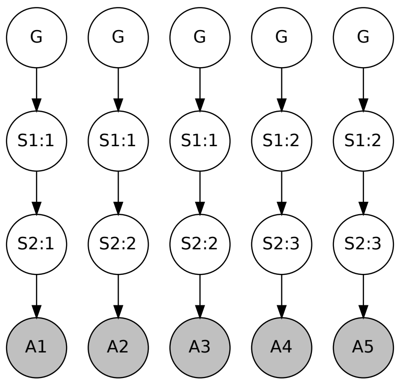
Once the CHMM has been formed, it is trained using the MONROE corpus — a SHOP2 plan-library which has been constructed using the same principles as described above, albeit on a larger scale [19]. This provides transition and emission probabilities, along with a prior goal distribution. The *forward algorithm* is then applied to predict the next n observations, based upon the trained CHMM and any previously observed states. In practice this leads to unexceptional results, so the CHMM is augmented to include a bigram model as in their earlier work. While this produces accurate results the average plan length of the MONROE corpus remains very low, at 9.5 actions per plan.

Hong

Hong's approach [89] to the problem of goal recognition was the first known system based on planning methodologies. It extends the concept of a *plan graph*



(a) A hierarchical plan as exemplified by output of the SHOP2 planner [127].



(b) A complete set of linear paths through the original hierarchical plan, as produced by Blaylock and Allen's work [20].

Figure 2.1: A hierarchical plan, in which the root node represents the overall goal of the plan, and each child node is a decomposition of its parent node. Leaf nodes in the tree are terminal actions in the final plan. For example, the root goal *G* can be decomposed into the achievement of sub-goals *S1:1* and *S1:2* in left-to-right order. Were figure 2.1b to be used in a CHMM, an additional set of edges would be present on all nodes other than those representing actions. These would connect each node/hidden state to its immediate right-hand neighbour, with downward facing arrows representing the *emissions* of the hidden state.

[23] — a structure which represents every possible path from an initial state to a goal — to produce a *goal graph*. Unlike other previous models of recognition, Hong's model is used in an *a-posteriori* context. That is, the system can only determine the agent's goal after the entire plan has been observed. The work is aimed at *explaining* an agent's plan, rather than predicting what the plan will be and what its purpose is.

While the model is not predictive, is does share a strong similarity with other models of plan and goal recognition, namely that recognition is based on eliminating inconsistent goal candidates. Hong does this through constructing the goal graph — a data structure used to track facts that may be goals, and eliminate any which are no longer consistent with the observed plan.

First, the goal graph is initialised with a single fact layer containing all facts present in the initial state. After each observation a new layer is added containing all facts in the newly updated world state. Facts which were deleted by the observed action are also recorded in their negated form. In addition to this new fact layer, a parallel layer is added containing all goals consistent with the previous observations. A goal G is consistent if it was added[10] by the last observation O_t or if there is a causal link between O_t and an unseen action O_{t+1} which achieves G. The goal which the system proposes as the final hypothesis is that which is *most consistent* with the observed plan. This is simply the goal which has the most actions contributing to its achievement.

As the goal graph has no probability or inference associated with it, the system scales extremely well, with the number of goals that can be considered in the tens-of-thousands. In addition to this, there is no assumption of optimality in the plan observed.

Ramírez and Geffner

Since Hong's work in 2001, the only known work which has made a similar use of planning techniques in recognition is the work presented herein and that of Ramírez and Geffner [145, 146, 147]. Like this work (although development of these two research avenues was independent), the problem is formulated as one of determining the final goal of an agent as it transitions through the state-space. However, there are also many differentiating factors which have a significant impact upon the way the system is used and the domains tackled.

Ramírez and Geffner's (R&G) early work [145] is analogous to Kautz and

[10]The same principle applies to negated facts.

Allen's model of PR, in that it is non-probabilistic and hypotheses are produced as a side-effect of eliminating inconsistent goals. R&G remove the assumption of a plan-library and instead model the problem as one of *domain theory* with respect to a known set of goals. The principle here is that the agent will always select the optimal or near-optimal plan for achieving the goal. Therefore, if an action is observed which is non-optimal with respect to a given goal, it can be eliminated from hypothesis consideration. R&G use both an optimal [77] and modified suboptimal planner [88] to determine the distance to each goal considered after an observation.

In their follow-up work [146], R&G move to a probabilistic model and relax the assumption of rationality in the agent. Here, after each action O_t is observed, a standard domain-independent planner is used [161] to compute a plan from the new belief-state to each possible goal. A second plan is then generated for every goal in the same goal-space, with the exception that O_t is removed from the action set used to generate a plan. This enables the recogniser to determine whether a goal becomes easier or harder to achieve once O_t is observed, versus if it is never observed.

Finally, R&G's work culminates in tackling the problem of *uncertainty* in observing an agent by applying their previous research using a partially observable Markov decision process (POMDPs) [96]. Previously, they consider the world as deterministic and fully-observable, but here they adopt Charniak and Goldman's view of recognition as inference under uncertainty [38]. Uncertainty arises from having only a partially observable representation of the subject's state and stochastic action effects. Baker *et al.* [12] have also investigated the use of planning in modelling agent recognition through the use of a MDP and POMDP, particularly in the context of a theory-of-mind.

Like Hong's work, the goal-space in R&G's work is generated at runtime, meaning a plan-library is not needed. However, while it is technically feasible to consider any number of goals, only a trivial number are evaluated in the associated literature. In particular, the repeated use of a POMDP planner [25] could make large-scale goal-spaces difficult to accommodate.

Prior Work Conclusions

Previous research in both plan and goal recognition has attempted to find a unified model of recognition which can cover all facets of the problem. However, all of these models can trace their roots back to Kautz and Allen's original definitions, in that the assumption of a plan-library being present remains (with the exception of those planning-based approaches given [12, 89, 145, 146, 147]). While there have been attempts to automatically generate such structures, this research has not progressed or been widely applied. Recently, Keren *et al.* have taken an alternate view of this problem, by using planning techniques to assist in the *design* of GR problems. Instead of generating plan libraries for a given problem, they attempt to reduce the number of *non*-unique plans for each potential goal, thus simplifying recognition [99]. However, this offline process requires that $O(n^2)$ optimal plans be generated (where n is the number of possible goals), reducing application to domains with a relatively small goal-set/domain size.

In contrast, the planning-based approaches given suffer from tractability due to complex modelling techniques. This makes rapid deployment of such systems difficult without some human intervention.

2.4 Motivation for a New Model of Recognition

The previous section presented an overview of the major models and accepted approaches to both plan and goal recognition. This section details the various motivating factors for a new model of goal recognition which are not addressed (or only partially addressed) by prior work.

Before each motivating factor is presented, it is useful to have a concrete example on which to draw these motivations from.

2.4.1 The Need for Recognition

Consider the following example, in which we observe a subject leaving their house and travelling around a city before returning home. First, the subject prepares breakfast at 08:00, exits the house and drives due East for several city blocks before turning North and driving to a supermarket. They enter the supermarket during which time they cannot be observed, and emerge 10 minutes later carrying a bag. They then drive to a factory and again cannot be observed until they exit at 17:05. They then drive West to a school where another person enters the car, and both return to the driver's house. They are then observed preparing a meal and consuming it.

At its simplest level and *a posteriori*, this scenario poses one question: what was the agent's plan? A human can easily infer that it is someone going to work and picking up their child on the route home, but this requires years of prior knowledge of how the world works. To a machine there are countless possible plans. Was the goal simply to return home? Was it to be at the factory? Was stopping at the supermarket required before they could visit the factory? Perhaps the goal was simply to be in the car at several points during the day, or to remove the second person from being at the school. All of these are viable hypotheses given the evidence presented, and it is here that the plan recognition problem lies.

At a more complex level, this problem demonstrates partial-observability (consuming breakfast was not observed, merely the preparation); non-optimal plans (there may have been a shorter route to the supermarket); conjunctive, intermediate and ordered goals (the supermarket had to be visited before dinner was prepared) and finally, potentially demonstrates multiple agents (does the child execute their own plan or are they merely a part of the parent's overall plan?). Harder still is the problem of the plan being presented *incrementally* and a hypothesis required after each observation. Finally, there must be an assumption that *any* valid combination of goals is possible and that *any* length of plan, P, can achieve it (where $optimal(P) \leq |P| < \inf$, and $|P|$ is unknown).

This set of constraints closely mimics those of a real-life recognition scenario, although further assumptions such as adversarial agents are also possible. Yet, many of these are not reflected in prior work, with each researcher selecting a subset of the problem to tackle. While some of these are often required to make the problem tractable, some assumptions, such as the length of an agent's plan, can overly constrain the problem to the point that real-world deployment would be unlikely to succeed.

With this example in mind, the following sections will discuss the various ways in which previous plan and goal recognition models fail to address the complete scenario. There is also a discussion of other, non-model related aspects of previous work which prevents wider acceptance of the field.

2.4.2 Tractability of Plan Libraries

As detailed above, the use of plan libraries is the standard approach to plan or goal recognition. Yet it is accepted within the recognition community that the construction and availability of such structures is a hindrance to evaluation and deployment. While it is possible to generate such libraries at runtime [111] or in an offline context [92], the vast majority of recognition research uses libraries constructed by hand, possibly by a domain expert, or from recordings of observed plan-traces [6, 18, 37, 60, 70, 97, 111].

While the construction of a plan-library is an expensive process, the plans contained within it are also crucial to many recognition implementations. Kautz and Allen's [97] original model assumes that the library is complete, with respect to all plans which can possibly be observed. Were this model to be applied in the above example in which the plan designer has not accounted for visits to the school, no hypothesis produced could take this into account.

In this context, *complete* is a misleading term. The library can only be truly complete if it contains all possible plans for all possible goals — something which is not only intractable but can potentially be infinite (if loops are allowed). Thus, a complete library can only be generated if the meaning of this is constrained in some way, e.g. only optimal plans. On the other hand, completeness includes all *extraneous* plans too. While models such as Charniak and Goldman's [38] allow any plan to be a sub-plan of another, others such as Lesh and Etzioni's [111] require that extraneous plans be eliminated from the library to prevent poor hypotheses being presented. In the example scenario this may mean ignoring plans in which the parent visits the supermarket twice, having forgotten to pick something up on the first visit. In the case of the library-free approaches such as Ramírez and Geffner's, the system would need to know which subset of the potential goal-space to consider as the true goal in order for results to be produced in a reasonable timeframe. Such restrictions may make using a library feasible, but they limit the ability for the system to accommodate uncertainty and redundancy in the agent's plan. Similarly, the techniques Keren *et al.* use to increase the uniqueness of a plan (and thus the associated goal), require that optimal plans be computed — essentially constructing a plan-library by side-effect [99].

2.4.3 Initial Probability Distribution

In probabilistic inference, the quality of the *initial probability distribution* across the goal and plan-space can result in high (or low) quality hypotheses being produced during the early stages of recognition. Like library design, the selection of these probabilities is often left to domain-experts or a *uniform probability distribution* is applied [38, 70, 147]. Alternatively, some systems use machine learning and prior observations to estimate these values [1, 18] .

While the latter approach is a useful step towards increasing the generalisation of a recognition system, it requires that many training examples be observed first. Additionally, just as a domain-expert may not be available to construct a plan-library, a suitable (possibly annotated) training set may not be available either. In the absence of such training data or prior knowledge, it is better for the recogniser to provide a *per-problem* initial distribution, based upon the *structure* of the problem being examined.

As stated previously, extraction of domain and problem features at runtime is common in the field of planning. Detection of invariants [16, 51, 53, 64, 65, 149], goal-orderings and landmarks [103, 115, 138, 148] are performed at runtime on the problem at hand in order to speed up search or improve plan quality. Examples of domain-independent feature extraction are unknown in the field of recognition, despite many problems (including those commonly cited in the literature) demonstrating an underlying goal structure from which probabilities can be derived.

2.4.4 Plan and Goal Abandonment

Much of the work discussed previously has been evaluated in a research context or is purely theoretical, with no deployment in the real-world. In such research-based evaluation, agents often have only a single goal or plan, which they pursue without deviation until completion. Unfortunately, this assumption is rarely applicable when considering agents in real-life recognition deployment, in which plans and goals may change over time. Therefore, it is crucial for a recogniser to be able to detect that a plan or goal which was previously being pursued has now been *abandoned*.

Little prior work exists in the field of plan and goal abandonment, with only Geib and Goldman providing an explicit model for this [61]. Building on their earlier PHATT model [70], they estimate the probability of a goal[11] G having been abandoned as a function of the number of timesteps since action O_s was observed, where O_s is known to precede the unobserved action O_{s+1} in the HTN plan for G. If O_{s+1} has not been observed after $k > 0$ further observations, then the probability that the plan/goal which it is associated with has been abandoned increases for every observation which follows O_s. The boundary for classifying a plan as abandoned is dictated by a parameter, ϵ, where $\mathrm{P}_{abandon}(G) \leq \epsilon \leq 1$.

The concept of abandonment is a stronger case of *negative evidence*, which Geib, Goldman and Millar discuss in their earlier work [70]. They define this to

[11]As described in Section 2.3.1, the PHATT model uses HTN plan representations, which have a single root node representing the goal. Thus, there is a known plan for each goal which can also be considered abandoned.

be the failure to observe an expected action. For instance not seeing the subject in the example scenario consuming breakfast, despite preparing it. In this case, an explanation may be that they are running late and have realised they do not have time to eat. However, more generally the failure to observe can be explained by a failure in the sensor model, or that the agent may have abandoned their plan (in this case, a plan in which they eat breakfast).

Geib and Goldman's later model can successfully represent the abandonment of goals, but requires a prior knowledge of action-orderings (that is, a *plan*). Further, it requires an additional calculation for the probability of the plan being abandoned. Without this and a suitable value for ϵ, the system cannot work. Thus, a further motivation for a new model of recognition is that plan abandonment be an integral part of the model, rather than a separate or additional feature of an existing model.

2.4.5 Standardisation

The example scenario outlined above uses natural language to describe the problem and actions observed. In this form, the underlying representation is determined by the reader. For instance, plans may be hierarchical or flat in nature; goals can be individual or conjunctive; or the adult and child may be viewed as separate agents (although the concept of "adult" and "child" are themselves inferred by the reader!).

This reader-specific representation mirrors that of prior recognition research. As a scientific field, there is little standardisation or compatibility between recognisers due to a lack of standardised input or output. While this prevents cross-testing of problems by external parties, the problem is further exacerbated by the absence of a common evaluation metric.

Problem Formulation

Given the problem domain at hand, each researcher implements their own model of the problem. While innovation in solving the recognition problem is of course to be encouraged, ambiguity in how the problem is *defined* can lead to different assumptions about the problem and thus differing solutions.

A strong motivating factor for this work is that input should be in an existing standardised form, such that other researchers can evaluate the system and construct new test domains with relative ease. Output can also be presented in a standardised form.

Evaluation Metrics

Closely linked to the issue of a standardised problem representation, is the method in which recognition systems are evaluated. Carberry noted in 2001 that there was still no common evaluation schema for plan or goal recognition [35], something which remains the case today. There has been some recent speculation that a common metric could come from holding a community-wide competition [71, 132], similar to the International Planning Competition [117], although there are fears it may stifle research and alienate some areas of the community.

Indeed, it could be argued that by nature, recognition does not lend itself towards a common evaluation metric. The range of problems which can be classed as “recognition” is so diverse that comparisons are often impossible. It may even be that in order to elegantly tackle the issue of metrics, a common input/output formalism must first be agreed upon.

2.5 Summary

The work presented in this thesis mimics many of the advances in planning outlined in Section 2.2, but applies them in the context of goal recognition. As the following chapters will show, heuristics can be used to determine which goal the subject is working towards, without prior knowledge of the plans which achieve these. Further, domain analysis can provide a means for *bootstrapping* the recognition process, in order that hypotheses can be produced without observing the agent’s behaviour.

Chapter 3

A New Model of Plan Recognition

3.1 Introduction

Chapter 2 outlined ways in which planning and recognition can share a common model. This chapter presents a new formal model of goal recognition, by introducing a common planning-recognition problem representation. By modelling recognition as one of *inverse planning*[1], it becomes possible to apply historical and current planning research.

The purpose of this formalism is that it negates the need for a plan library to be present, and instead frames the problem as one of an agent traversing a *state-space* — that is, executing a plan. The formalism presented in this chapter is based upon the assumption of rationality in both the executing agent and observer (although the former will be relaxed to allow suboptimality in plans).

As the problem representation put forth in this chapter is based upon optimal heuristics, it is ultimately intractable in the real-world due to its complexity [32]. To guarantee optimal heuristic estimates, the entire state-space must be explored, which results in exponential processing time or memory requirements. However, Chapter 4 provides several means by which the problem can be *relaxed* to obtain a usable solution.

[1]The term *inverse planning* is borrowed from Baker *et al.* [12].

3.1.1 Overview

Section 3.2 formalises a common model between planning and recognition, before Section 3.3 demonstrates how heuristics can be applied to the problem. Here, recognition becomes a process of eliminating inconsistent goals (in line with Kautz and Allen's model [97]), with respect to optimality in the agent's plan. This consistency-based approach is then relaxed in Section 3.4, which removes the assumption of rational agents through introducing a probabilistic representation. This form allows *inference* to take place during the recognition process. Finally, Section 3.6 details how the final model relates to previous work in the field of recognition, and any prominent features of the model which have not been previously enumerated.

3.2 A Base Planning and Plan Recognition Formalism

Many of the terms which appear in both planning and plan recognition literature can be described by a common formalism. For example, both problems use literals, actions, states and goals which all share the same definition. Moving beyond this, more abstract concepts such as the *state-space* and *goal-space* can also have a common definition, which allow the potential for performing recognition (or planning) using research from both fields. This section will enumerate and define the various components required to perform such a task.

3.2.1 Core Shared Terminology

Conceptually, planning is the problem of constructing a path through the *state-space* of a problem — a graph of all states which are reachable from the initial state through action transitions. As Section 1.3 described, the same structure exists in recognition, with the observer wishing to determine the final state the agent will end in.

The base model of a state-space is well known in planning, wherein the world is represented as a set of boolean propositions (also referred to as *facts*), popularised by the STRIPS formalism [49], with no metric or temporal aspects and the world is fully observable and transitions are deterministic. This frames the problem as a *classical planning problem.*

Definition 1. Classical Planning Problem
A classical planning problem is a tuple $\pi = \langle F, A, I, G \rangle$, where F is a set of literals

in first-order logic and A is the set of all grounded actions in the domain. $I \subseteq F$ is the initial state for the problem and G is the goal $G \subseteq F$, $G \neq \emptyset$. Each action $a \in A$ is a triple $\langle a_{pre}, a_{add}, a_{del} \rangle$, where $a_{pre}, a_{del} \subseteq F$ are the preconditions and delete effects of a, respectively, and $a_{add} \subseteqq F$ are the add effects. Each action $a \in A$ has an associated cost, $c(a) = 1$.

In classical planning the closed-world assumption applies. That is, the application of (valid) action $a \in \pi(A)$ which is applicable in state S, will always lead to a *reachable* state S', and any literal which is not known to be true is assumed to be negated, with no further literals or actions added after problem instantiation. A state is reachable if it can be achieved through a series of actions applied in order from the initial state. An action is valid if it can be applied through a sequence of state-transitions from the initial state. Such states are *sound* — they contain no *mutually-exclusive* facts. Note that every action must have at least one add effect, as the negation of a literal is not considered a valid precondition or goal.

Two facts are mutually-exclusive (mutex) if it is impossible for both to be true in the same state. The set of all mutex relations associated with each planning problem, $M(\pi)$, is a mapping of each fact $f \in \pi(F)$ to a further set of literals, $f \rightarrow M(f) \subset F$, where $M(f)$ may potentially be the empty set.

These properties allow the planning problem to be represented as a directed graph, known as the *state-space*, in which vertices are states and edges are the actions which transform states into *successor states*.

Definition 2. State-Space
The state-space of a problem is a directed, possibly cyclic, graph $\mathbb{S} = \langle V, E \rangle$, which represents all action transitions and resulting states in a planning problem π. Each vertex $v \in V$, is a state, and each edge $\langle S, S' \rangle \in E$ is an action, $a \in \pi(A)$, for which $a_{pre} \subseteq S$ and $S' = (S \setminus a_{del}) \cup a_{add}$.

Definition 3. Reachable State
A state $S \in \mathbb{S}$ is *reachable* if there exists a finite path of totally-ordered action transitions from the initial state $\pi(I)$ to S.

While theoretically possible, enumeration of the state-space for all but the simplest of problems is intractable. Yet, in the case of recognition the problem is even harder, as the state-space is of limited assistance in determining the final goal. This stems from the fact it is unlikely an agent's goal will be the achievement of a state $G = S \in \mathbb{S}$. Rather, it will be a *subset* of a state, $G \subseteq S \in \mathbb{S}$. Just as states exist within the state-space, goals are deemed to exist within the *goal-space*.

3.2.2 The Goal-Space

The state-space represents how an agent can move through intermediate, causally-linked states in order to achieve its final goal. The *goal-space* is a superset of this where instead of vertices representing states, they represent an explanation of why the previous action was applied. That is, the "goal" of applying an action is to achieve a subset or all of the literals within the successor state. For example, if an action a is executed in state S, it is because the goal[2] was not wholly contained within S, but is within S'.

Definition 4. Goal-Space
The goal-space of a problem is a directed, possibly cyclic, graph $\mathbb{G} = \langle V, E \rangle$, in which each node $v \in V$ represents a state comprising a set of non-mutually-exclusive facts $v \subseteq \pi(F)$. Each edge is an action $a \in \pi(A)$. For an edge to exist between vertices $\langle u, u' \rangle$, the preconditions of a must be at least partially applicable in u, such that $(a_{pre} \cap u) \subseteq a_{pre}$ (where a_{pre} and a_{del} can be the empty set, but a_{add} cannot). For each partially or wholly applicable action, u will have $2^{|K-1|}$ outgoing edges, where $K = (u \setminus a_{del}) \cup a_{add}$. The associated outgoing vertex set is equal to $\bigcup_{i=1}^{2^{|a_{add}|-1}} \wp_i(u \setminus a_{del}) \cup a_{add}$, where $\wp_i$ is the ith member of the associated *power set*. If only full states are considered as goals, $\mathbb{G}$ is equivalent to $\mathbb{S}$ — or more generally $\mathbb{G} \supseteq \mathbb{S}$.

Like the state-space, the notion of causally-linked action transitions is retained, but in a relaxed manner. Here, an action is considered applicable if the goal is merely a partial subset of the action's preconditions. While in the context of the state-space these transitions would be illegal as the state need not include all an action's preconditions, for the goal-space the agent may have only wished to achieve a single fact as the intermediate goal, with any other achievements being *side-effects* of the action. The goal-space is not intended to represent legal action transitions, but rather an agent's logic in selecting an action or plan. For example, in Figure 3.1c there is a single initial state $I = \{X\}$. If the agent executes action B, they will have done so in order to achieve one of the successor goal

[2]Here, the term "goal" merely refers to the immediate reason for applying an action. In practice, there may be several transitions through the state-space (i.e. a plan). Thus it may be appropriate to think of members of the goal-space as *intermediate goals*, unless the plan is known to have finished.

states $\{U, XU\}$. If the goal were simply to have X true, no action should have been executed. By moving to another state within $\mathbb{S}$, the agent has indicated that the goal was not wholly contained within the predecessor state.

By defining the planning problem and goal-space, the *goal recognition* problem can now be formalised.

Definition 5. Goal Recognition Problem
A goal recognition problem is a triple $\gamma = \{\mathbb{G}, I, G^*\}$, where $\mathbb{G}$ is the goal-space of the problem; I is the initial state and $G^* \in \mathbb{G}$ is the true goal of the agent moving through $\mathbb{G}$, which is unknown to the observer.

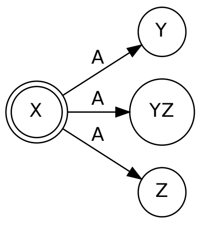
While planning is defined as movement through the state or goal-space to a known goal, the equivalent goal recognition problem is viewed as the same movement with no knowledge of $\pi(G)$ (where $\pi(G)$ and $\gamma(G^*)$ are equal). Of course, in the case of planning the agent's movement through $\mathbb{G}$ has a known end point, while in recognition this terminal state is unknown.

Viewing the Goal-Space as an Observer

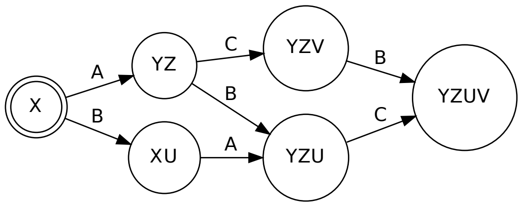
While the goal-space represents a *deterministic* transition model to the executing agent, for an external observer it will be *non-deterministic*[3]. This is for the simple reason that only the agent knows *why* it transitioned into state S', while the observer must *infer* this. More importantly, only the agent knows which goal-state $G' \in \mathbb{G}$ it has entered. For example, if the subject's goal-space contains only three facts $\{X, Y, Z\}$ and a single action A which deletes X and adds $\{Y, Z\}$ such as that in Figure 3.1a, it is impossible to determine which of the successor goals is the true goal. Thus, at any point in time the subject's goal can be considered to be one of the $2^k - 1$ current *belief-goals* $G^B \in \mathbb{G}_t$, where $k = |S|$ is the number of literals which comprise the current state $S \in \mathbb{S}$ and $\mathbb{G}_t$ is the goal-space at time t. It is also possible to consider the situation where no observations will be made, indicating that $G^* \subseteq I$.

While accounting for non-determinism in the model may seem to increase the complexity of the problem, much of this uncertainty only remains until the next action is observed. At this point some of the belief-goals present before the observation may be eliminated from the set of possible agent goals depending on the form of the action. If the observed action has no delete effects, then *all* previous belief-goals can be ignored, because the addition of new facts indicates that the

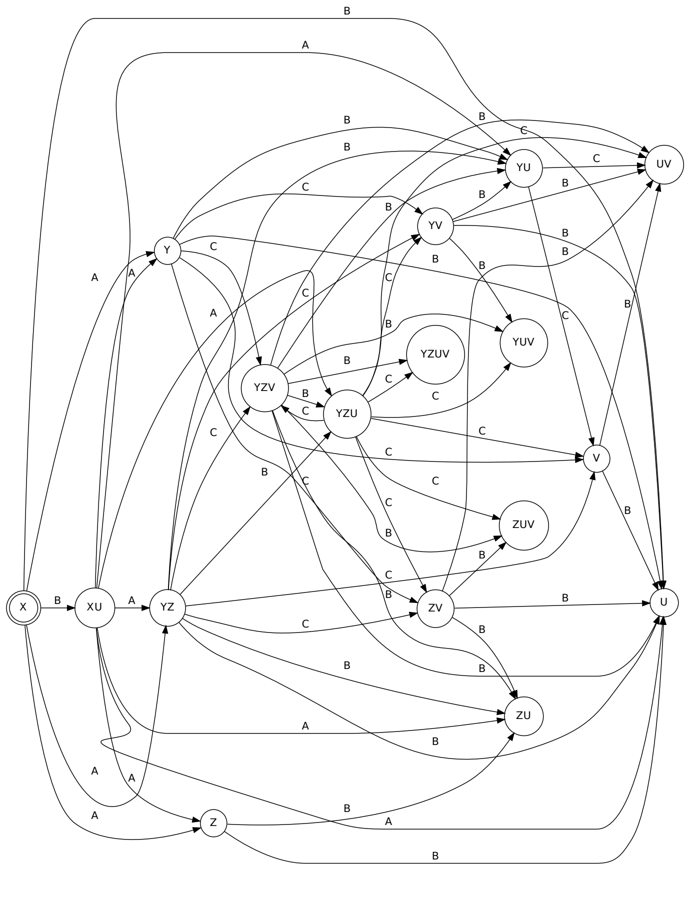
[3]Note that in a fully-observable environment, the *state-space* remains fully deterministic for both subject and observer.



(a) An example of non-determinism in the goal-space



(b) A fully enumerated state-space



(c) The goal-space representation of Figure 3.1b

Figure 3.1: Representations of various spaces within the model.

goal is at least partially contained within the newly added literals. However, if delete effects are present, then only those belief-goals which are a partial superset of the delete effects can be eliminated, because the removal of a fact is enough to indicate that it is an unwanted goal[4]. To clarify this point, consider the following example in which the subject being observed moves through the (fully enumerated) states-space shown in Figure 3.2 and associated goal-space given in Figure 3.3.

The agent exists in an extended version of the previous example world, which now has predicates $\{X, Y, Z, U, V\}$, and three actions $A = \langle X, \{Y, Z\}, X\rangle$, $B = \langle \emptyset, U, \emptyset\rangle$ and $C = \langle Y, V, \emptyset\rangle$, which are of the form $a = \langle a_{pre}, a_{add}, a_{del}\rangle$. The agent still starts in state X, and is observed executing action A. As before, this transitions the agent from X to one of several possible belief-goals representing the agent's potential reason for executing A (i.e. they wanted to achieve Y, Z or YZ). The agent then executes action B which leads to the set of belief-goals becoming $\{U, YU, ZU, YZU\}$, before finally executing action C. This causes the final set of belief-goals to be $\{V, YV, ZV, UV, YZV, YUV, ZUV, YZUV\}$.

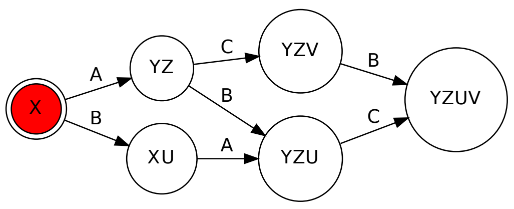
Once the first action has been observed, and fact X deleted, all hypotheses containing X can be eliminated from $\mathbb{G}$, as it is no longer achievable. After the second observation, U is added to the state, which means that only goals containing this should be considered. This is also true after the addition of V in the third observation. The question of which of the belief-goals represents the true goal will be considered in Section 3.3.

The example above shows the complexity which arises from the uncertainty involved in viewing the goal-space as an observer. However, while the example maintains a *consistent* set of belief-goals (that is, the plan so far is "consistent" with each belief-goal being the goal), it is a naïve model as many can be ruled out. For instance, V can be removed from the final set of hypotheses because selecting C as the second action would have been a shorter plan to achieve solely V. Thus, the model assumes that the agent is *rational*, and will always take the optimal path to the goal.

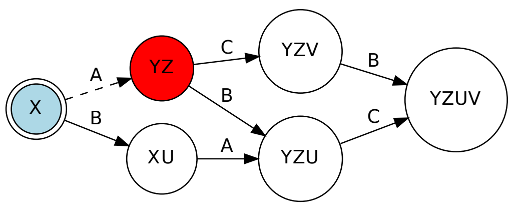
Definition 6. Rational Agent
An agent is *rational* within a state-space $\mathbb{S}$, if for their goal G^* and initial state I, all plans executed achieving G^* from I are guaranteed to be of minimal cost.

The implications of this assumption are clearly strong, and heavily limit the

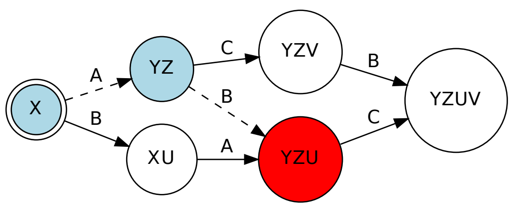
[4]The impact of this assumption will be discussed further in this chapter.



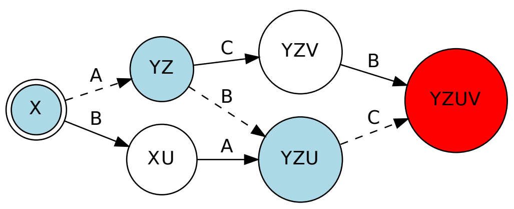
(a) Initial state-space



(b) After action A is observed.

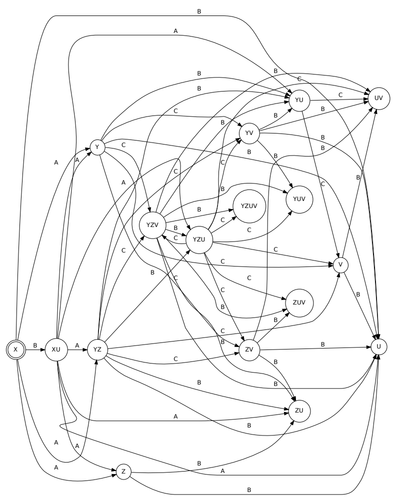


(c) After action B is observed.

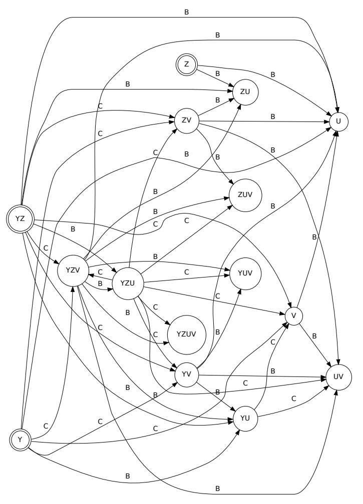


(d) After action C is observed.

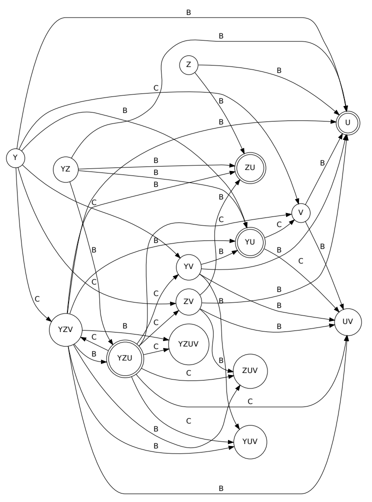
Figure 3.2: The agent's movement through the example state-space during execution of plan $P = \langle A, B, C \rangle$. The initial state is highlighted with a double-circle, with the current state is highlighted with a black border. The action selected to transition out of the current state is shown as a dashed arrow. States which have been passed through remain shaded.



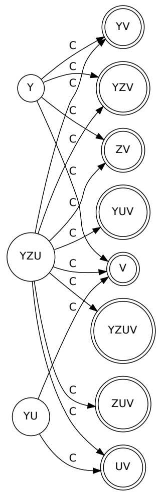
(a) Initial goal-space



(b) After action A is observed.



(c) After action B is observed.



(d) After action C is observed.

Figure 3.3: The agent's movement through the example goal-space during plan execution, corresponding with actions taken in the state-space shown in Figure 3.2. Belief-goals are highlighted using double-circles. Figure 3.3a shows the complete goal-space at problem instantiation, with all optimal paths to each goal from state X. After the first observation is processed, all states in the previous goal-space which can no longer be reached by an optimal path from the original initial state are eliminated. As further observations are processed the goal-space reduces until (in this example case) only belief-goals remain.

application of the model in a real-life scenario. Luckily, this assumption can be relaxed to allow for suboptimal agents, as will be discussed in Section 3.4. For now, it serves as a useful aid in explaining the principles of the model presented.

3.2.3 The Plan-Space

The goal-space of a problem — while monotonically decreasing in nature if a rational agent is assumed — remains constant throughout plan execution, with no new goals being added to it. However, associated with every planning problem π is a further graph representing the *plan-space* of the agent. This graph is the set of all possible plans which can achieve the agent's goal G^* from their initial state. In recognition, since there is no knowledge of G^*, this space must be expanded to also include all plans to all goals in $\mathbb{G}$. This is somewhat analogous to a fully-enumerated representation of a traditional library-based PR system with a *complete* set of plans, such as Kautz and Allen's work [97, 98].

Definition 7. Plan
A plan is a pair $P = \{I, O\}$, where I is the initial state of the agent $I \in \mathbb{S}$, and O is a series of totally-ordered actions $\langle O_1 ... O_n \rangle$, or the empty set.

While unintuitive, empty plans must be considered as valid, as Definition 1 does not state that $G^* \notin I$. Therefore, it is possible for the agent's goal to be already achieved in the initial state. In such cases, the state and goal-spaces will never reduce as no observations will be processed. However, this does not prevent generation of a hypothesis, as I is always a member of $\mathbb{G}$ at problem instantiation.

In the context of recognition, each plan has an associated set of belief-goals G^B. This is the set of belief-goals existing after the action sequence has executed, one of which will represent the true goal $G^* \in G^B$.

Definition 8. Plan-Space
The plan-space $\mathbb{P} = \{S \to \mathbb{G}\}$ is a finite[5] mapping of all optimal plans from the current state $S \in \mathbb{S}$ to all reachable goals in goal-space $\mathbb{G}$. Plans within $\mathbb{P}$ may be of length $0 ... \tau$, where $\tau = \frac{\varnothing(\mathbb{S})}{2}$ is the radius of $\mathbb{S}$ which represents the longest possible acyclic plan. Zero-length plans indicate that the goal is achieved in S.

The plan-space may alternatively be thought of as a series of paths through both the state and goal-spaces. However, if plans are allowed to overlap and

[5]In a STRIPS context with an optimal agent, the path through $\mathbb{G}$ should always be finite, as looping plans would be impossible because the plan will terminate as soon as every required goal literal has been met by a state transition.

intersect, a graph structure becomes a more natural method of representation. Alternatively, Figure 3.4 shows a representation of the plan-space which includes a representation of the goal-space. Regardless of this, all plans exist in the same form in the state, goal and plan-spaces, with only the observer's perception affecting the deterministic or non-deterministic movement through each space.

This section has presented a common representation for goal recognition and planning, based on the assumption of *rational* agents which have *optimal* plans. The *state*; *goal* and *plan-spaces* were also introduced and detailed, with each modelling the movement of an agent as it executes its plan. In particular, the *goal-space* was described as a means of modelling the fact an agent will execute an action in order to achieve a subset of the successor state. That is, one of these subsets is the agent's *intermediate goal*. Until now, only intermediate goals have been considered as hypotheses, however, the true task of goal recognition is to determine the *final goal*.

Of course, in order to achieve the final goal, an agent must first achieve n intermediate goals by executing actions, where $n = |P| - 1$. The process of selecting an action to apply is *planning*. The next section discusses how techniques for action selection related to planning can be utilised in goal recognition, in order that the final goal can be deduced.

3.3 Heuristic-Based Goal Recognition

Given a reachable goal G, a rational agent will always find a minimal-length plan which achieves this (or in the case of an unreachable goal, it will know this is impossible). As the state-space is a graph, the planning process is one of searching through this structure to find the shortest path from the initial state to a goal-state. This is done through application of a heuristic, h, and search algorithm such as A*. If the agent is using an *admissible* heuristic, then estimates are guaranteed to never exceed the true the distance to the goal.

Determining which action is the optimal choice can be achieved using an optimal heuristic, denoted h^+. In the context of action selection, a heuristic returns a value $h^+(G) \in \mathbb{Z}^+$ corresponding to the number of further actions required to achieve the goal. It is possible [121] for the heuristic estimate to be a real value $h(G) \in \mathbb{R}^+$, but in a classical-planning model such as that presented, only integer values are considered.

In planning, heuristics are used to guide search through the state-space. An

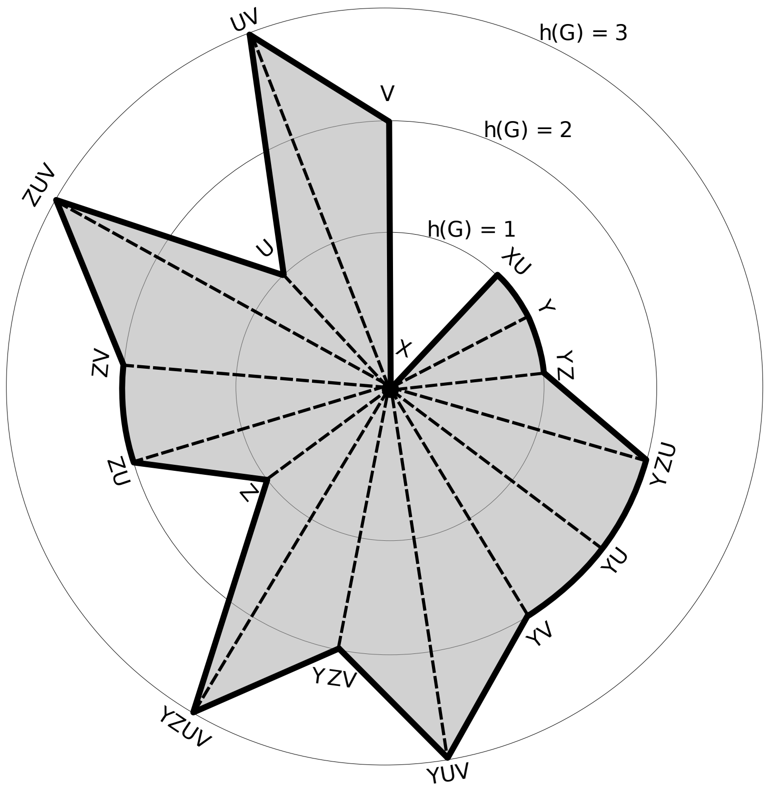


Figure 3.4: An alternative visual representation of the plan and goal-spaces from the problem described in Section 3.2.2, starting from the initial state. The goal-space is represented as the perimeter of the polygon, with every reachable goal represented. The distance from the centre indicates the number of steps required to achieve the goal. Only optimal plans are included, and are represented as dashed lines emanating from the centre of the polygon to the specific goal they achieve. While not explicitly shown, it is possible for there to be multiple instances of the same plan. For example, ZUV and $YZUV$ are both achieved through plan $\langle B, A, C\rangle$. X has no associated plan as it is true in the initial/current state.

optimal plan can be derived only if there is always an action which lowers $h^+(G)$ after application. This process is shown in Algorithm 1, wherein each successor of the current state is expanded, and used to re-test the heuristic. The successor state which has the lowest heuristic estimate has the action which achieved it added to the plan (see Figure 3.5).

The type of heuristic used in plan generation indicates the class of planning which it belongs to. *Domain dependent* planning uses heuristics which are tailored to a specific problem or class of problems and provide highly accurate and reliable results. Alternatively, *domain independent* heuristics are designed to provide good results across any problem and can still produce optimal plans. Naturally,

Algorithm 1 getOptimalPlan(I, G^*)

```
$S_{curr} := I$ {Set current state to initial state}
$P := \langle\rangle$ {Create empty plan}
**while** $G^* \nsubseteq S_{curr}$ **do**
  $h_{best} := getEstimate(S_{curr}, G^*)$ {Get estimate to goal from current state}
  $A_{app} := getApplicableActions(S_{curr})$ {Get actions applicable in $S_{curr}$}
  $A_{best} := \{\}$ {Actions which give the best successor states}
  **for all** $a \in A_{app}$ **do**
    $S_{next} := apply(S_{curr}, a)$ {Get successor state}
    $h_{succ} := getEstimate(S_{next}, G^*)$ {Re-estimate using $S_{next}$}
    **if** $h_{succ} < h_{best}$ **then**
      $A_{best} := \{\}$
      $append(A_{best}, a)$
      $h_{best} := h_{succ}$
    **else if** $h_{succ} = h_{best}$ **then**
      $append(A_{best}, a)$
    **end if**
  **end for**
  **if** $|a_{best}| = 0$ **then**
    **return** Fail: No optimal plan
  **end if**
  $a_{chosen} := rand(A_{best})$ {Randomly select from best action set}
  $append(P, a_{chosen})$
  $S_{curr} := apply(S_{curr}, a_{chosen})$
**end while**
**return** $P$
```

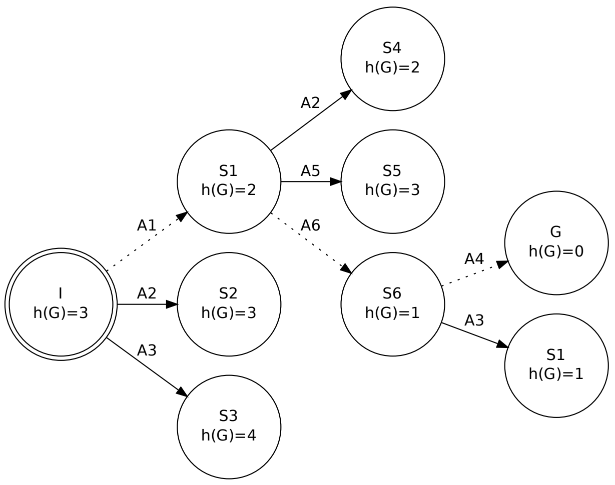


Figure 3.5: Using a heuristic to guide search through the state-space. The initial state is highlighted with a hollow border, and any state containing the goal with a black border. Successor states are expanded after each action application and the distance to the goal, G, estimated. Successors which have the lowest estimate have their associated action chosen, until a successor state which contains G is found.

it is this latter class which is of relevance in performing domain-independent goal recognition.

3.3.1 Observing Planning

Now consider how Algorithm 1 operates. After a new state has been entered (or at initialisation), each applicable action in the state is individually applied to construct a set of new successor states. The heuristic estimate from each of these states to the goal is then computed. In an optimal context, the best action to choose is that which has the lowest heuristic estimate from its corresponding successor state. Therefore, $h^+(G)$ always lowers after each observation until it becomes equal to zero.

This behaviour, whereby the agent always selects the action that lowers $h(G)$, is visible to external observers with one caveat — they are unaware of what G is. Note that while they may not know that G is the agent's true goal, they may be *aware* that G exists and therefore able to compute $h(G)$ themselves, although this value may be different to that which the agent has computed (due to differing heuristics). Thus, goal recognition can be modelled as the observation of the agent's movement through $\mathbb{G}$, in order to determine which goal is being *converged* upon. This goal or set of goals, will therefore contain the agent's true goal.

Definition 9. Heuristic Convergence
For a goal-space $\mathbb{G}_t$ and plan P, a heuristic is said to be *converging* on a goal $G \in \mathbb{G}_t$ if $h^+_t(G) < h^+_{t-1}(G)$, for all timesteps $1 \leq t \leq |P|$, where $|P|$ is the unknown length of the plan. After each observation the size of the goal-space will be reduced by at least one, such that $\mathbb{G}_t \subset \mathbb{G}_{t-1}$, given that there are no actions with both empty preconditions and effects.

With the model of planning which has been outlined above, goal recognition becomes a process of observing an agent moving through a goal-space and deducing a set of final belief-goals, one of which represents their goal. Heuristics — possibly the same as the agent is using[6] — can be used to get the estimate to each member of the goal-space, and *eliminate* any goals for which $h^+_t(G) \geq h^+_{t-1}(G)$. This is shown in Algorithm 2.

While heuristic convergence in planning is concerned with only the work required to achieve the goal from the current state, in recognition the *work*, $W(G)$, which has been expended in achieving the goal is of greater interest. Given that there is no knowledge of the agent's true goal in the goal-space, a recogniser must calculate the heuristic estimate to *all* goals $G \in \mathbb{G}$. For an optimal heuristic and rational agent, the estimate $h^+(G)$ will decrease monotonically until plan termination if $G = G^*$. After each observation, a new estimate of the work remaining to each goal in the goal-space is computed and compared against the estimate at the previous timestep. If the application of the observed action lowers the heuristic estimate (or achieves) the goal, it is said to have has *contributed* towards the goal.

The difference in the heuristic value at time $t = 0$ and the current time $t = |O|$ is therefore an estimate of the work expended on achieving goal G (Equation 3.1). As the goal-space is guaranteed to reduce after each observation (Definition 9), all goals $G \in \mathbb{G}_t$ will have the same value for $W(G)$. Equation 3.1 demonstrates this. Any goals which are no longer reachable after an observation are said to have had no work expended upon them — as by definition the goal cannot be one of these.

[6]The topic of agent and recogniser using the same heuristic is evaluated in Chapter 5.

$$W_{h^+}(G|O_t) = \begin{cases} h_0^+(G) - h_t^+(G) & \text{if } G \text{ is } \textit{reachable} \\ 0 & \text{if } G \text{ is } \textit{unreachable} \end{cases} \tag{3.1}$$

While all members of the goal-space at any timestep will have the same value of work associated with them, it is only the belief-goals which can be considered as candidates for a *hypothesis*.

To illustrate this, recall the example outlined in Figure 3.3. If this is now revisited using a heuristic-based approach to the elimination of goals, the size of the final set of belief-goals can be halved to $\{UV, YUV, ZUV, YZUV\}$. The reason for this reduction is that all other goals can be ignored due to there being shorter plans which can achieve them. Table 3.1 shows how heuristic estimates can be used to eliminate members of the goal-space whose distance has not lowered after the previous observation.

G	$h(G)$	$h(G\|A)$	$h(G\|B)$	$h(G\|C)$
X	0	-	-	-
XU	1	-	-	-
Y	1	0	-	-
YZ	1	0	-	-
YZU	2	1	0	-
YU	2	1	0	-
YV	2	1	-	-
YUV	3	2	1	0
YZV	2	1	-	-
$YZUV$	3	2	1	0
Z	1	0	-	-
ZU	2	1	0	-
ZV	2	1	-	-
ZUV	3	2	1	0
U	1	1	0	-
UV	3	2	1	0
V	2	1	-	-

Table 3.1: Heuristic convergence on a subset of $\mathbb{G}$ for the plan $P = \langle A, B, C \rangle$. All goals which cannot be achieved by an optimal plan after each observation are denoted as '-' and can be eliminated from $\mathbb{G}$. Note that X cannot appear with any goal other than U, as it not *reachable* if the first observation is anything other than B.

Algorithm 2 getNewGoalSpace($\mathbb{G}_t, S_t, O_t$)

```
$\mathbb{G}_{t+1} := \langle\rangle$ {Create successor goal-space}
$S_{t+1} := apply(S_t, O_t)$ {Get successor state}
for all $G \in \mathbb{G}_t$ do
    $h_t(G) := getEstimate(S_t, G)$
    $h_{t+1}(G) := getEstimate(S_{t+1}, G)$
    if $h_{t+1}(G) < h_t(G)$ then
        $add(\mathbb{G}_{t+1}, G)$ {If estimate has reduced, add to new goal-space}
    end if
end for
return $\mathbb{G}_{t+1}$
```

Historical Context

By using optimal heuristic estimates to minimise the set of belief-goals, the model presented becomes equivalent to Hong's work on determining the goal of an agent *a posteriori* [89]. Hong accomplishes this by tracking the number of observed actions which have contributed towards achieving the goal. An action contributes to a goal if it achieves the goal, or it is part of a series of *causal-links* which connect a previously-achieved literal to the goal.

The model outlined above and Hong's model can be considered equivalent only if the agent is rational. Under such assumptions, every action will be considered a causal-link in achieving the true goal. However, instead of tracking such links between actions, an optimal heuristic is used to achieve the same result. Of course, Hong can achieve this without such assumptions of optimality.

The assumption of rational agents also draws parallels with Kautz and Allen's library-based model [98]. Here recognition is classed as minimising the set of top-level goals which are consistent with the observed plan. Like Kautz and Allen's model, any belief-goals which are consistent are assumed to be equally probable, something which is rarely the case.

While there are similarities with Hong's work and the use of optimal heuristics in the previous section, Hong allows consistent goals to be ranked by the number of observed actions which have contributed towards it becoming achieved (that is, the number of causal-links the goal has). This allows the assumption of rationality to be removed, and introduces ordering of hypotheses. Ideally, the model presented above should also have the ability to rank hypotheses.

Ramírez and Geffner adopt a similar model in their earlier work [145], whereby the agent is again assumed to be rational. This consistency-based model allows them to produce the same output as Hong's work, albeit they use this in an online context.

The assumption that the agent being observed will be rational, and any heuristic used to evaluate the goal-space will be optimal, has been done to simplify the presentation of the initial model components. It has intentionally restricted any predictive capabilities to only allow hypotheses for the previous observation. Such hypotheses are equal to the current set of belief-goals which maximise $h^+(G)$, with no way to distinguish between goals of equal score. These issues are addressed in the following section by adapting the model to incorporate probabilistic reasoning with suboptimal agents.

3.4 Incorporating Probabilistic Inference

The model presented in the previous section, while potentially applicable, poses two key problems to deployment. The first is that optimality in agent behaviour rarely exists in the real-world. Agents — whether they be human or virtual — may derive an optimal plan, but it is far likelier that they will execute a suboptimal plan which achieves their goal in near-optimal time. The second is that all members of the set of consistent goals are equally likely. It is impossible to select one as being the most probable, despite most real-world goals demonstrating a *prior probability* of being the true goal. In addition to these, the model is not predictive — it can only produce hypotheses in an *a posteriori* context. As this section will demonstrate, all of these issues can be at least partially resolved if the model is adapted to become *probabilistic*.

The first step in this conversion is to place a probability distribution across the goal-space.

Definition 10. Probabilistic Goal-Space
A probabilistic goal-space $\mathbb{G}^P$ is a standard goal-space $\mathbb{G}$ with associated probability distribution ϑ in which every goal $G \in \mathbb{G}$ maps to a probability $\mathrm{P}(G) \rightarrow \vartheta(G) \in [0:1]$.

Before recognition begins, the goal-space can have an *initial probability distribution* imposed upon it. These prior probabilities can come from domain knowledge or can be computed at runtime using domain analysis. If no information about the domain can be extracted, a *uniform* distribution is applied such that $\mathrm{P}(G) = \frac{1}{|\mathbb{G}|}$, $\forall G \in \mathbb{G}$. For now, a uniform distribution will be assumed with non-uniform distributions considered later in Chapter 4.

In this probabilistic representation, the assumption that agents are rational is relaxed, as this would otherwise be identical to the original, non-probabilistic version. Instead, the agent is assumed to exhibit *bounded rationality*, in which the agent actively pursues the goal but may unintentionally execute unnecessary or unhelpful actions. The assumption that an optimal heuristic is available to the observer for measuring the distance to goals will be retained. This assumption will itself be relaxed in Chapter 4.

Definition 11. Bounded Rationality
An agent exhibits *bounded rationality* if they produce a plan $P = \langle O_1 ... O_n \rangle$ for goal, G^*, in which the *majority* of plan steps cause the goal to become heuristically closer. The set of *positive* observations, O^+, contains any action O_i for which $h_n(G^*) < h_i(G^*) \leq h_{i-1}(G^*)$, for plan steps $i = \{1...n-1\}$. This must be greater than the set of *negative* observations $O^- = P \setminus O^+$, for which $h_i(G^*) > h_{i-1}(G^*)$.

While the agent may potentially have an optimal plan, it is more likely the plan will be suboptimal but that, crucially, it will still achieve the goal without deviating wildly from the optimal plan. Actions which do not alter the distance to the goal are allowed, as are actions which *increase* the distance to G^*. These negative actions can be viewed as poorly-informed decisions made by the agent, and are tolerable, provided that at the end of observation, $|A^+| > |A^-|$. This is similar to Hong's premise of goal *consistency* when explaining the purpose of a completed plan [89].

Occasionally, previous work in recognition has allowed for suboptimal agents, albeit in a weaker form. In particular, Goldman *et al.* [70] allow agents to perform actions "for the sake of it". This translates to agents being permitted actions which have *no effect* on the achievement of a goal. For instance, if all goals are related to the movement between locations, an observation in which the agent reads a book will be of no consequence in reaching their destination[7]. Such actions are equivalent to the heuristic estimate for a goal remaining the same over n observations, $h_t(G) = h_{t+1}(G) = ... = h_{t+n}(G)$. However, if the agent does indeed demonstrate bounded rationality, an observation which does *not* reduce the estimate cannot be classed as irrelevant, as they may simply have taken a poor decision.

[7]It will be assumed that the agent is not referring to a map!

3.4.1 Heuristic Estimates as a Metric of Work Performed

Definition 4 describes the goal-space as a monotonically decreasing graph which is modified after each observation. Now that the agent is no longer assumed to be rational, the goal-space will only decrease in size if a goal is deemed to be unreachable. For instance, in the running example (Figure 3.1a, page 38), after action A is observed, fact X can never be re-achieved. This causes all goals which contain X to also become unreachable. All reachable goals cannot be pruned as it is always possible for a goal to be achieved using a suboptimal plan (which covers looping, potentially infinite plans).

Despite the goal-space now being constant (aside from any pruning of unreachable goals), the set of belief-goals at time t will remain as before, as these merely explain the reason for transitioning between states. The definition of *work* performed by a plan being equal to the difference between the initial and current estimates can also be retained. However, in this form values for work can be in the range of natural numbers $W(G) \in \mathbb{N}$, while it would be preferable to have a *normalised* value in the new probabilistic model.

To achieve this, Equation 3.1 can be modified to derive the *maximum likelihood* (ML) of the observations seen thus far contributing towards a specified goal. This is denoted as the number of observations which have resulted in the goal becoming *nearer*, versus the total number of observations. A goal is classed as becoming nearer at time t if $h_t(G) < h_{t-1}(G)$. Conversely, it is classed as having become *further* away if $h_t(G) > h_{t-1}(G)$. Equation 3.2 outlines this new definition.

$$W_{ML}(G|O_t) = \frac{\sum_{i=1}^{t} \begin{cases} 1 & \text{if } h_i(G) < h_{i-1}(G) \text{ and } i > 0, \\ 0 & \text{otherwise} \end{cases}}{|O|} \tag{3.2}$$

Equation 3.2, in conjunction with the now potentially suboptimal agent, provides both a means of ranking members of the current belief-goals and a basic approach to *predicting* the final goal. Table 3.2 shows the values of W_{ML} associated with the running example. If G^* is any goal containing U and V, the plan $\langle A, B, C \rangle$ is optimal as $W_{ML} = 1$. However, if the goal is YV this previous plan will be suboptimal, as executing action B will have no effect on the heuristic estimate for this goal. Note that after each observation, all remaining reachable facts must be still considered as possible goal candidates, as the length of the agent's plan is unknown. For instance, while goals containing literals U and V

G	$W_{ML}(G\|A)$	$W_{ML}(G\|B)$	$W_{ML}(G\|C)$
X	**0**	**0**	**0**
XU	**0**	**0**	**0**
Y	1	**0**	**0**
YZ	1	**0**	**0**
YZU	1	1	**0**
YU	1	1	**0**
YV	1	0.5	0.667
YUV	1	1	1
YZV	1	0.5	0.667
$\boldsymbol{YZUV}$	1	1	1
Z	1	**0**	**0**
ZU	1	1	**0**
ZV	1	0.5	0.667
ZUV	1	1	1
U	0	**0**	**0**
UV	1	1	1
V	1	0.5	0.667

Table 3.2: Values of work performed after each observation. In the first column, the true goal is highlighted in bold. Entries in other columns which are bold have a zero-value and are no longer reachable, in accordance with Equation 3.1.

have maximum probability, other goals with lesser probabilities cannot be eliminated as there may be a further series of observations which are helpful towards their achievement.

With this said, using the ML score as a sole means of estimating the final goal is acceptable only under the condition that all goals are considered equally-probable at problem initialisation. As this is unlikely to be the case, the ML score can be used instead in the computation of a Bayesian posterior probability. This allows the prior probability of each goal to be used in addition to the observed evidence. Note that for now these prior probabilities are assumed to be provided by an external source. Chapter 4 describes an automated means of extracting these on a relaxed form of the complete model.

3.4.2 Predicting the Final Goal Using Bayesian Inference

Bayesian inference has a long history in recognition [1, 30, 38, 50, 91, 93], and is a natural choice for determining whether observed evidence matches up with a hypothesis (in this case, a member of the goal-space).

After each observation, O, every member of $\mathbb{G}^P$ has its probability updated according to the standard Bayesian inference formula (see Equation 3.3), with the work expended on the goal, W_{ML}, offering a convenient *likelihood function* (Equation 3.4). This results in a *posterior* probability, $\mathrm{P}(G|O)$, which indicates the prospect of G being the goal given the plan seen thus far and any prior probability of it being the goal. In this form the denominator $\mathrm{P}(O)$ captures all goals which are mutually-exclusive with G, rather than the simple value of W_{ML} which only implicitly captures this. In the complete goal-space model presented in this chapter this property is somewhat moot, however it will be invaluable once the relaxed model is introduced in Chapter 4.

$$\mathrm{P}(G|O) = \frac{\mathrm{P}(G)\mathrm{P}(O|G)}{\mathrm{P}(O)} = \frac{\mathrm{P}(G)\mathrm{P}(O|G)}{\sum \mathrm{P}(G_i)\mathrm{P}(O|G_i)} \qquad \forall G_i \in \mathbb{G} \tag{3.3}$$

$$\mathrm{P}(O|G) = W(G|O) \tag{3.4}$$

If the example outlined previously is now revisited using Bayesian probability, Table 3.3 is derived. If a uniform initial probability distribution is used, the same set of goals (those containing UV) will be selected for the hypothesis set. While it is good to know that the original results have not been destroyed by the move to Bayesian inference, there is still no way to distinguish between results with the same score. To do this, a non-uniform initial probability distribution must be applied. Table 3.4 shows the difference in results if an initial distribution which favours goals YZV and $YZUV$ is used. Both goals remain the most probable after observation A, but after the second observation, $YZUV$ becomes the most probable, as fact U has been added by the observation.

The assumption of having access to a good initial distribution is perhaps too convenient for the running example, given that the agent is indeed trying to reach $YZUV$. Fortunately, Bayesian probability allows the model to accommodate a poor or naïve initial probability distribution being used instead.

For example, consider the example presented in Table 3.5, wherein goal Z has been incorrectly favoured in the initial distribution. As observations are processed, the distribution changes to favour those goals which are supported by the current evidence.

Encountering an uninformative initial probability distribution is analogous to monitoring for plan and goal abandonment. In both cases prior evidence suggests that a particular goal is being pursued, followed by new evidence which no longer

G	$P(G)$	$P(G\|A)$	$P(G\|B)$	$P(G\|C)$
X	**0.059**	0	0	0
XU	**0.059**	0	0	0
Y	**0.059**	**0.067**	0	0
YZ	**0.059**	**0.067**	0	0
YZU	**0.059**	**0.067**	**0.1**	0
YU	**0.059**	**0.067**	**0.1**	0
YV	**0.059**	**0.067**	0.05	0.063
YUV	**0.059**	**0.067**	**0.1**	**0.188**
YZV	**0.059**	**0.067**	0.05	0.063
$\boldsymbol{YZUV}$	0.059	**0.067**	**0.1**	**0.188**
Z	**0.059**	**0.067**	0	0
ZU	**0.059**	**0.067**	**0.1**	0
ZV	**0.059**	**0.067**	0.05	0.063
ZUV	**0.059**	**0.067**	**0.1**	**0.188**
U	**0.059**	**0.067**	**0.1**	0
UV	**0.059**	**0.067**	**0.1**	**0.188**
V	**0.059**	**0.067**	0.05	0.063

Table 3.3: Probabilities for the example plan using Bayesian inference with a uniform initial probability distribution and optimal heuristic. Items in bold are the maximum value for their respective column.

supports the previous hypothesis. For instance, goal Z **is** supported by the first observation, but not by the remainder of the plan.

Section 2.4.4 discussed Geib and Goldman's work on goal abandonment [61] as one of explicitly measuring the probability of an action in a plan never being seen, given that the action which is known to precede it is successfully observed. However, in reality this model is targeted at plan abandonment only — as the abandonment of a goal is tied directly to the plan which achieves it. The heuristic-based approach presented exists at a lower level than this, where plans are unknown. That is, Geib and Goldman's representation of the distribution across $\mathbb{G}$ is hidden beneath the distribution across the plan-space, $\mathbb{P}$.

The running example is sufficient to demonstrate the basic principles of the model presented, but its simplicity masks a subtler problem which is unique to non-library-based recognition. Consider a hypothesis Hyp, representing the current belief of the agent's terminal goal. As the model is now probabilistic, it is natural for any hypothesis to be the goal which has the maximum probabil-

G	$P(G)$	$P(G\|A)$	$P(G\|B)$	$P(G\|C)$
X	0.013	0	0	0
XU	0.013	0	0	0
Y	0.013	0.014	0	0
YZ	0.013	0.014	0	0
YZU	0.013	0.014	0.019	0
YU	0.013	0.014	0.019	0
YV	0.013	0.014	0.009	0.008
YUV	0.013	0.014	0.019	0.023
YZV	**0.4**	**0.411**	0.280	0.227
$\boldsymbol{YZUV}$	**0.4**	**0.411**	**0.561**	**0.682**
Z	0.013	0.014	0	0
ZU	0.013	0.014	0.019	0
ZV	0.013	0.014	0.009	0.008
ZUV	0.013	0.014	0.019	0.023
U	0.013	0.014	0.019	0
UV	0.013	0.014	0.019	0.023
V	0.013	0.014	0.009	0.008

Table 3.4: Probabilities for the example plan using Bayesian inference with a favourable initial probability distribution and optimal heuristic. Items in bold are the maximum value for their respective column.

ity $Hyp = \arg\max \mathrm{P}(G), G \in \mathbb{G}^P$. However, a hypothesis produced using this principle ignores a critical component which is implicitly present in library-based recognition — the length of the agent's plan.

3.4.3 Hypotheses and Estimating Plan Length

In planning, heuristic convergence upon the goal (that is, $h(G^*) = 0$) indicates that the search process can now stop. In library-free goal recognition where there is no knowledge of the length of the agent's plan, it is impossible to determine if the current state which follows any observation is an *intermediate* or *terminal* state (and hence, whether it contains the goal)[8].

For instance, in Figure 3.3 (page 41), if the agent's complete plan was $P = \langle A, B \rangle$ for the goal $G^* = \{Y, Z, U\}$, there is no way to know that $\{Y, Z, U, V\}$ should be eliminated from the potential set of goals. In this case (and given equal initial probabilities), the heuristic has converged by the same amount for both

[8]Unless the state which has been entered is a *dead-end*, where $|\mathbb{S}_t| = |\mathbb{G}_t| = 1$, meaning it is impossible to transition out of the state.

G	$P(G)$	$P(G\|A)$	$P(G\|B)$	$P(G\|C)$
X	0.006	0	0	0
XU	0.006	0	0	0
Y	0.006	0.006	0	0
YZ	0.006	0.006	0	0
YZU	0.006	0.006	**0.1**	0
YU	0.006	0.006	**0.1**	0
YV	0.006	0.006	0.05	0.063
YUV	0.006	0.006	**0.1**	**0.188**
YZV	0.006	0.006	0.05	0.063
$\boldsymbol{YZUV}$	0.006	0.006	**0.1**	**0.188**
Z	**0.9**	**0.911**	0	0
ZU	0.006	0.006	**0.1**	0
ZV	0.006	0.006	0.05	0.063
ZUV	0.006	0.006	**0.1**	**0.188**
U	0.006	0.006	**0.1**	0
UV	0.006	0.006	**0.1**	**0.188**
V	0.006	0.006	0.05	0.063

Table 3.5: Bayesian probabilities for the example plan when provided with a poor initial probability distribution. Goals which are no longer reachable are essentially pruned from the goal-space by having a zero value for $W(G)$ (see Equation 3.1).

goals. This means that at any point during the recognition process, the recogniser cannot be sure if the previous observation achieved the agent's goal, or whether there will be an unknown number of further observations, $\varepsilon \in \mathbb{Z}^+$, which can be used to further refine the set of hypotheses.

To state this in a more concise form, the challenge is no longer to select only the goal which is the most probable given the observed plan, but to further refine this set by estimating which goals are achievable given the *unseen* future observations. This is equivalent to a probability distribution across the plan-space from the current state, where probabilities represent the likelihood of each plan being the agent's true plan. In turn, this will cause a further distribution across the goal-space, derived from the associated plan. However, instead of there being only a single goal-space distribution representing the final goal, it is possible to construct a further distribution for each unseen observation at time b, where $1 \leq b \leq \varepsilon \leq \tau$, and τ is the radius of the state-space (see Definition 8). As such, hypotheses can be *bounded* by placing a value on the estimated number of steps remaining, which limits the size of the goal and plan-spaces.

3.4.4 Bounded Goal Hypotheses

In an environment where the agent is rational, and provided that there have been fewer than τ actions observed so far, the number of observations remaining cannot be determined, but will have an upper limit of $\tau - |P|$. In environments where agents merely demonstrate bounded rationality, knowledge of τ would be of limited use, as the agent's plan can potentially be suboptimal or looping. Here, an approximation of this, τ_{max}, can be constructed by taking the maximum estimate to any goal in the goal-space, using an optimal heuristic. In other words, the agent is assumed to be optimal from "now" until the end of observation. Alternative methods of approximating this include purely analytical means [99, 116]; by utilising a plan-library [20], or constructing a model of agent behaviour [1].

$$\tau_{max} = \arg\max_{G} h^{+}(G) \quad \forall G \in \mathbb{G} \tag{3.5}$$

With a value assigned to τ, it becomes possible to produce *bounded* hypotheses. These are goals which the agent will have to achieve prior-to the final goal. In other words, while the plan P which is being executed will be to achieve some final goal in $\mathbb{G}$, there will be $|P| - 1$ goals which must be achieved prior to this that are mutually-exclusive with G^*. The agent's bounded goal associated with each of these timesteps is the difference between the state at time t and preceding state $S_t \setminus S_{t-1}$. Alternatively, the bounded goal at time t is equal to the effects of observation O_{t-1}.

Definition 12. Bounded Goal Hypothesis
A *bounded goal hypothesis*, Hyp_c^b, is a goal $G \in \mathbb{G}, G \neq G^*$ created at time c on the premise that $G = S_{c+b} \setminus S_c$ is the goal to be achieved at time $c + b$, where $0 \leq c < |P| - 1$ and $1 \leq b \leq |P|$.

Bounded hypotheses differ from final goal hypotheses in that they represent *only* the individual facts in state S_{c+b} which are required to change from state S_c, in order to reach state S_{c+b+1}, or achieve the goal in S_{c+b}. Therefore, while intermediate hypotheses can be any member of the goal-space, bounded hypotheses are a subset of this. As the observer has access to an optimal heuristic, it is trivial to derive all possible bounded hypotheses for all goals in $\mathbb{G}$, when combined with the plan-space $\mathbb{P}$. Chapter 4 explores how such hypotheses can be used in practice.

This chapter has thus far presented a model of goal recognition based upon the use of heuristics in modelling the agent's behaviour. Motivation for this has stemmed from the belief that a plan-library should not be required in order for recognition to take place. Yet, it would be folly to ignore such resources if they *are* available. It is not in dispute that libraries offer a rich source of information about agents, plans, action orderings and goals. The next section demonstrates how such structure can be integrated into the model in order that goal recognition can be improved.

3.5 Library Integration

The previous description of the goal-space offers a *complete* model of all goals which can be achieved within the problem. If a library, Γ, is available, $\mathbb{G}$ can be modified in one of two ways. The first is that the recogniser can simply replace the standard goal-space with the library goal-space, $\mathbb{G}^{\Gamma}$. This will likely be a highly refined subset of the original goal-space, from which any extraneous goals have been removed. Secondly, the library may also offer information on the prior probabilities of goals. Under the assumption that the agent's goal is indeed part of $\mathbb{G}^{\Gamma}$, the resulting goal-space is potentially minimal and contains no irrelevant entries. While this is ideal, the constraint that the agent's goal must be a part of the library is too strong for real-world deployment, or for scaling beyond trivial goal-spaces. A second, better approach, is to *integrate* the information contained within the library, with the standard goal and plan-spaces.

As $\mathbb{G}$ should be complete, all goals within $\mathbb{G}^{\Gamma}$ should already be present. Therefore, of greater interest is the initial probability distribution across $\mathbb{G}^{\Gamma}$. This, as with the contents of the library, will most likely have been optimised using prior observation of the domain or expert knowledge.

Indeed, the presence of a plan-library can even negate the requirement of a heuristic. For instance, if all plans are known, their length can be used as a lookup table in place of a heuristic. Of course, as such completeness is unlikely and has been argued against throughout this chapter, it would be better to view this lookup table as a more *informed* heuristic. The next chapter shall demonstrate that the optimal heuristic which has been discussed in this chapter is rarely a possibility in the real-world.

3.6 Model Features and Historical Context

3.6.1 Library-Free Recognition

The model detailed above shows how heuristics can be applied to goal recognition by representing the problem in a planning formalism. The model is *complete* in that all possible goals are considered a member of the goal-space. It is *sound* as any hypothesis produced should be a valid goal candidate at the time of hypothesis generation. Should any goal become unreachable during observation, it cannot be put forward as a hypothesis candidate.

The speed and scalability of the complete model given is dictated by the size of the goal-space. The runtime is linked to the complexity of the underlying heuristic used in observation, which will need to be called (at most) $|\mathbb{G}|$ times, per observation. The following chapter will look at how this truly manifests itself in the relaxed model.

By incorporating existing plan libraries, the model is somewhat backwards-compatible with prior research. Access to an existing library would most likely make recognition more accurate, and potentially reduce runtimes.

3.6.2 Relationship with Prior Art

Aside from the primary features of the model given above, many of the requirements put forth in previous PR literature are also covered. In particular, Goldman, Geib and Miller (GG&M) [70] provide extensive criteria for the recognition system to adhere to. This section shall briefly enumerate the most important of these and how the heuristic-based model accommodates them.

- **Negative Evidence** — The model uses heuristics to detect those facts/goals which have become "closer" after an observation, and assigns a higher probability weighting to these literals. Conversely, if the heuristic indicates that an observation has not assisted in making a goal become closer, the associated goal-probability is reduced. The latter can be viewed as a form of *negative evidence*, which GG&M put forward as a means of detecting both missed observations and goal abandonment [60, 61, 70]. While GG&M model abandonment as an additional computation which is linked to the probability of seeing observation O_{k+1} given that O_k has been observed, the heuristic-based model would naturally degrade the associated probability over time. That is, there is no explicit assumption of causality between successive observations, but that this is implicitly modelled by the heuristic estimate not reducing (or increasing) over time. The weaker case of an agent executing an action "for the sake of it" is also covered by the heuristic-based mode, as this simply translates to the heuristic estimate for all goals remaining the same across observations.

- **Pending Set** — Another of GG&M's model features is to include the state of the world, rather than relying purely upon observations. This allows a pending set of applicable actions to be available at any time during recognition, with the next observation coming from this set[9]. The heuristic-based model also incorporates the notion of state and therefore of the pending set.

- **Partially-ordered/Interleaved Plans** — In library-based recognition, partial-orderings between observations and multiple concurrent plans must be explicitly accounted for during modelling. In the heuristic-based model (given that the goal-space is complete), these orderings are irrelevant. Provided that each concurrent plan does not interact with the heuristic estimates for each respective goal (e.g. one lowering the distance to a goal, whilst the other increases it), the associated probabilities will be unaffected. If the observer has an optimal heuristic, partial-ordering of actions is also irrelevant, as the estimate will be accurate regardless of when the action is observed.

- **Plan/Goal Abandonment** — Detecting whether an agent has abandoned their goal has been largely ignored by the PR community. It is assumed that the agent will either always complete their original plan [70, 97], or that the "abandoning" of the goal is actually incorporated in the original plan [5]. Only Geib and Goldman have tackled this issue [61], by incorporating additional logic into their existing library-based model of PR [70].

  The heuristic-based model of recognition implicitly allows modelling of abandoned goals, by treating the original and subsequent plans as a form of suboptimality on the part of the agent. While Geib and Goldman must consider whether the actions observed have a match within the set of possible plans being pursued, the heuristic-based model naturally shifts probabilistic weights towards the goals which are currently being pursued. For example,

[9]Only guaranteed if the world is fully-observable.

once the agent has switched their goal (prior to achievement), the heuristic estimate to the new goal will begin to decrease, while the estimate to the old goal will increase[10], resulting in the probability of the new goal increasing over time.

Outwith the work of GG&M, others have proposed specific types of goals, which will be treated differently by executing agents than a typical "final" goal achieved in the last state. Motivated by safety in robotic/assistive plan execution, Weld and Etzioni proposed *dont-disturb* and *restore* goals [160]. In the former the goal should never be achieved (such as a software agent deleting files), while the latter indicates that the goal which is true in the initial state must also be true at the end of plan execution, as it may have been negated during execution.

Bacchus and Kabanza later incorporated these goal types in their work [7], renaming them as *safety* and *maintenance* goals. The model presented herein allows for both of these goal types to be recognised, albeit without formally classifying them. Bacchus and Kabanza's terminology is used as they also define two further goal groups which are of relevance. Note that the recogniser is unaware of these definitions, and treats all goals equally.

- **Classical Goals** — These are simply goals which must be achieved in the final state such as those described in the previous sections. Their previous values during plan execution are irrelevant as long as the fact is achieved at plan termination.

- **Safety Goals** — In a domain where the agent is assumed to demonstrate bounded rationality, safety goals will have fixed probabilities throughout observation. For every goal in $\mathbb{G}$ which contains a safety goal $G_s \subset G \in \mathbb{G}$, the associated posterior probability will be equal to $\mathrm{P}(G \setminus G_s|O_t)$. That is, while achievement of the safety goal is not assisted by O_t, it cannot be ruled out as being negated in the future (as the fact this is a safety goal is unknown to the recogniser), meaning it and any goals which it forms a part of must remain valid goal candidates.

- **Maintenance Goals** — Once negated, these goals are treated in the same manner as *classical* goals, in that any movement by the agent towards their achievement will be reflected in the associated posterior probability.

[10]Note that this is will only be the case once the estimate to the new goal has reduced by a greater amount than the original goal.

- **Timing Deadlines** — Weld and Etzioni define these goals to be those which must be true by a certain time, k, after plan execution begins. Here, these would be treated as classical goals which are achieved early in the plan (most likely prior to termination). While *timed deadlines* could be viewed as a form of *bounded* goal as described in Section 3.4.4, the explicit detection of achievement times these is not considered in this work.

3.7 Chapter Summary

This chapter has presented a new model of recognition as a process of estimating an agent's goals based upon heuristic estimates, in a planning-based problem representation. The features of the heuristic-based model have been enumerated in detail, and their relationship to previous models of recognition explained.

The original model given in Section 3.3, which is comparable with previous work [89, 97, 145], was relaxed to accommodate agents which demonstrated bounded rationality, which moved the model to a probabilistic form in line with previous beliefs on how the PR problem must be represented [38].

While the agent was no longer assumed to be rational, the assumption that the observer had access to an optimal heuristic was retained. The following chapter shall instead look at how the model can be relaxed to accommodate suboptimal heuristics, thus allowing tractable usage of any derived system.

Chapter 4

IGRAPH — A Library-Free Implementation of Goal Recognition

4.1 Introduction

Chapter 3 described in detail a model of goal recognition which uses heuristics to eliminate parts of the goal-space in order to determine and predict the agent's goal. Any implementation of this model would require a full enumeration of the state, goal and plan-spaces — something which is infeasible for all but the simplest of problems. Section 3.4 relaxed the assumption of the agent being *rational*, in order that suboptimal plans could be considered. However, the further assumption of an optimal heuristic, h^+, being available to the observer was retained. Unfortunately, this assumption is itself intractable in the real-world, with the derivation of any optimal heuristic having the same time and space complexity as the underlying problem [32].

Due to this intractability, any implementation of the model must be *relaxed* to make the problem solvable in reasonable time. As a comparison, *planning* is PSPACE-complete in its most expressive form, but simple relaxations [17] such as the *delete-list* relaxation [28, 88] allow the problem to become at most NP-hard for optimal heuristics [32]. If *suboptimal* heuristics are used instead, the problem can attain polynomial complexity and becomes tractable. This chapter will demonstrate that the same principles can be applied in the context of recognition.

4.1.1 Overview

This chapter introduces several relaxations to the model presented in Chapter 3, which enable recognition to become tractable on unseen domains. The implementation, IGRAPH (Intermediate Goal Recognition with A Planning Heuristic), successfully performs goal recognition on a variety of problems. Unlike other goal recognisers, IGRAPH is intended to be rapidly deployable. That is, it should be possible for a non-expert user to perform recognition with only a basic description of the domain. Knowledge of the agent's most probable goals and motivations cannot be assumed, meaning no prior knowledge can be encoded (such as prior probabilities).

To achieve this, a standard and widely accepted input language has been used (PDDL 2.1 [55]), which also allows existing domains to be used in evaluation. Section 4.2.1 provides an overview of the PDDL language, and also introduces a second formalism, SAS$^+$, which can be derived from and encodes a different (but ultimately equivalent) representation of the PDDL problem. IGRAPH combines these representations to allow techniques from both formalisms to be applied. Both representations are defined along with any relevant associated data structures. Additionally, this section also describes a technique for constructing an explicit, graph-based representation of the observed plan, something which is normally taken for granted in plan/goal recognition literature, where plan libraries are available.

Section 4.3 introduces several relaxations to the existing model which enable tractable goal recognition to be performed. The primary relaxation presented is to allow suboptimal heuristic estimates for members of the goal-space. This incurs further relaxations on aspects of the new model, such as the goal-space and reachable goals. Section 4.3.2 discusses a method for extracting conjunctive hypotheses from these newly relaxed goal-spaces, without loss of completeness. Several heuristics are introduced in Section 4.3.3 which allow a trade-off between accuracy and computation time, while Sections 4.3.5 and 4.3.6 describe how the Bayesian likelihood function can be modified from a maximum-likelihood approach to one more suited to a relaxed model. Aspects of the complete model which are implicitly present in the relaxed model are given in Section 4.3.7 and 4.3.8, followed by discussion of how goal abandonment can be detected in Section 4.3.9.

Work on the *automatic* derivation of prior probabilities using only domain structure is presented in Section 4.4, before the features of the relaxed model are summarised in Section 4.5.

4.2 Cross Domain Goal Recognition Using A Standardised Input Formalism

As Chapter 2 explained, planning and recognition share many similarities in their respective representations. Section 3.2 presented a formal base model of this crossover and noted that the difference between performing planning and recognition in this model is merely the utilisation of the heuristic. Thus, both planners and recognisers can be constructed from the same code-base and data-structures. This section demonstrates how the existing and widely adopted planning input formalism can be applied to recognition problems.

4.2.1 Utilising An Existing Planning-Based Representation

The use of first-order, propositional logic to represent problem components such as state and action effects is common throughout both the planning [23, 27, 49] and recognition literature [60, 89, 97]. IGRAPH is no exception to this, but unlike other work in recognition[1] it uses a standardised input language in the form of PDDL [55, 120]. In its simplest form, recent versions of PDDL closely resemble the STRIPS formalism [49] with no metric or temporal aspects in the model (also known as PDDL level 1). While these features are supported by modern day PDDL [47, 55, 67], IGRAPH is based solely upon the *classical* representation (as defined in Section 3.2). As such IGRAPH has the ability to perform recognition on all existing classical competition and benchmark domains which adhere to this formalism.

In addition to the PDDL representation which defines the problem being tackled, a second planning formalism can be derived using a suite of translation tools. This work [78], transforms the initial PDDL problem into a SAS$^+$ formalism [9], which extracts further domain knowledge.

PDDL

In PDDL, problems are split into two separate parts, which when combined form a planning-task, π, and the associated state-space from which the planning task

[1]With the notable exception of Ramírez and Geffner [145, 146, 147] who also use PDDL as an input language.

can be solved. The *domain* file contains a formal definition of object types, predicate and action templates and is similar to that used in previous recognition literature [18, 38, 109], where these would be referred to as *schema*. In planning, a schema is referred to as a *lifted* representation. It is this latter term which will be used for the remainder of this chapter.

The second input — the *problem* file — details a set of objects that exist in the world along with the initial and goal states. Each object has an associated *type*, and all facts in the initial and goal states must be *grounded*. A grounded fact is simply one in which all parameters have been assigned an object. Once parsed, these two files are combined to create a *grounded problem* which contains the initial state and the grounded fact and action sets. In the case of IGRAPH, the goal section is naturally ignored or not present in the problem file.

During the grounding process, invalid actions and facts can be created such as `(on crate1 crate1)`, or `(drive london london)`. These *unreachable* facts and actions can cause recognition difficulties in the form of *invalid* hypotheses. For example, a conjunctive hypothesis of the form `(and (on crate1 crate1) (on crate1 crate2))` is partially correct but unreachable by the agent.

The prospect of invalid and irrelevant facts being present in the recognition process is something which is often overlooked or ignored by previous work. Implementations which use a plan-library often follow Kautz and Allen's assumptions that the library will be *complete* [97], and that *soundness* is implied by all plans being legal and relevant. Others such as Lesh and Etzioni [109, 111, 112] make it explicit that there must be no possibility of illegal plans being present in the library, or recognition will fail.

If a plan-library is indeed used, this assumption is somewhat tractable, provided that the *source* of the plans contained therein is itself sound[2]. However, once the library is removed as in this work, valid goals must be derived from the domain description alone. This is problematic due to the grounding process having no knowledge that `(on crate1 crate2)` is valid, while `(on crate1 crate1)` is not, as both fit the required type constraints of the "on" predicate.

There are several methods for trying to resolve this issue, which vary the degree to which unreachable facts are filtered out versus analysis time. In this work, the output of Helmert's PDDL-to-SAS$^+$ translator is used as the source of reachable facts [78].

[2]Whether the source of plans be from human or machine is irrelevant, as long as their validity is provable.

The grounded problem generated from a PDDL representation is extremely uninformative in its initial form. No domain information is extracted or encoded in the input files, meaning any further data must be extracted through *domain analysis*. For example, mutually-exclusive facts are unknown and must be derived from the ground problem (although this is not a *necessary* step).

Of course, PDDL is merely one possible implementation of the model described in Chapter 3, albeit the most similar. A second formalism, SAS$^+$ [9], can encode the same information in a differing format. Recent work by Helmert [78] supports the translation of traditional PDDL problems into the SAS$^+$ formalism, and further, detects several useful problem features using domain analysis, which can be applied in the context of goal recognition.

SAS$^+$

The translation of a PDDL problem into a SAS$^+$ representation analyses the propositions and actions of the domain and problem file and converts them from first-order logic predicates into a set of *variables* $V = \{v_1 ... v_n\}$. Each variable represents a single object[3] specified in the problem file or derived from the domain description. The domain, D_v, of each variable, $v \in V$, represents a finite set of possible values equivalent to the set of facts which contain v as a parameter, or encapsulate the behaviour of an implicit variable. For instance, the PDDL representation may encode the individual facts `(at driver location1)`, `(driving driver truck)`, and `(at driver location2)`, while SAS$^+$ would encapsulate these as a single variable associated with `driver`, and domain $D_{driver} = \{\texttt{(at driver location1)}, \texttt{(driving driver truck)}, \texttt{(at driver location2)}\}$. Note that it is possible (and indeed, probable), that members of a variable's domain can appear in the domain of other variables too. In the above example, there would also be a variable for the truck, the domain of which would contain `(driving driver truck)`. This possibility and its implications on performing recognition in a relaxed environment will be discussed further in Section 4.3.1.

Definition 13. SAS$^+$ Problem
A SAS$^+$ problem is a tuple $\phi = \{V, A, I, G^*\}$, where V is a set of variables, A is a set of grounded actions, I is the initial state and G^* is a partial state representing the goal. States are represented as an assignment to all $v \in V$.

The primary reason for translating the original PDDL problem into this form is the production of *domain transition graphs* (DTGs) and a *causal graph* (CG).

[3]Note that a SAS$^+$ variable is *not* strictly equivalent to a PDDL object.

Each DTG represents the values which a variable v can take during plan execution, with every node being a single fact in which v appears as a parameter. Edges represent an action which once applied, will transition the current/outgoing value to the incoming value. Figures 4.1a and 4.1b respectively show the DTGs for a truck and driver in the DRIVERLOG problem given above[4]. The causal graph represents dependencies between variables within the problem domain. For example, the causal graph shown in Figure 4.1c shows that trucks are dependent upon drivers and vice-versa, while packages are only influenced by trucks.

Definition 14. Causal Graph
Given a SAS$^+$ problem ϕ, the associated causal graph is a connected, directed graph, $\{V, E\}$, where V is equivalent to the set of all variables $V = \phi(V)$ and E is a set of edges indicating an influence between two variables. A directed edge, $\langle u, v\rangle$, exists if $u \neq v$ and $\exists a \in \phi(A)$, such that $a_{add} \cup a_{del} \subset D_v$ and $a_{pre} \cup a_{add} \cup a_{del} \subset D_u$.

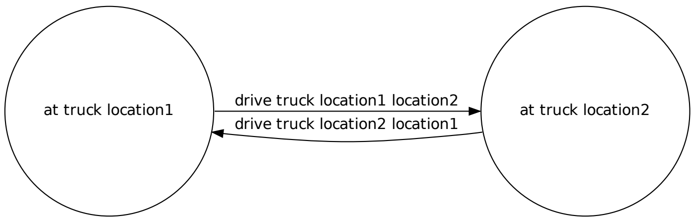
Definition 15. Domain Transition Graph
A domain transition graph is a directed graph which represents the legal transitions of a variable, v, through a SAS$^+$ problem ϕ. The vertex set, D, is equivalent to the domain of the variable, and the edge set, E, is comprised of actions which transition between vertices. An edge, $\langle d, d'\rangle$, exists if $d \in a_{pre} \cap a_{del}$, and $d' \in a_{add}$.

Further to the production of these graphs, information on sets of facts which are *mutually-exclusive* with one another is also produced during translation. However, this information is not complete, meaning not all mutexes are guaranteed to be detected. This, along with the possibility of illegal facts groundings discussed above, has implications for *hypothesis generation* — namely that of unreachable goals or goal conjunctions. These are explored in Section 4.3.2.

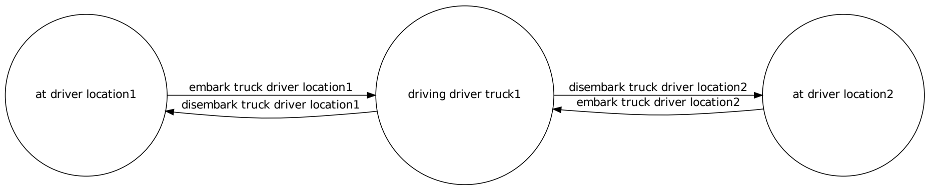
4.3 A Relaxed Model of Goal Recognition

Using the two formalisms defined above, it is now possible to present a *relaxed* model of the formalism presented in Chapter 3. Relaxations take the form of using suboptimal but domain-independent heuristics and reducing the scale of the goal-space, such that it becomes tractable. *Completeness* can be preserved, but at the cost of *soundness*. That is, the agent's true goal will always be contained

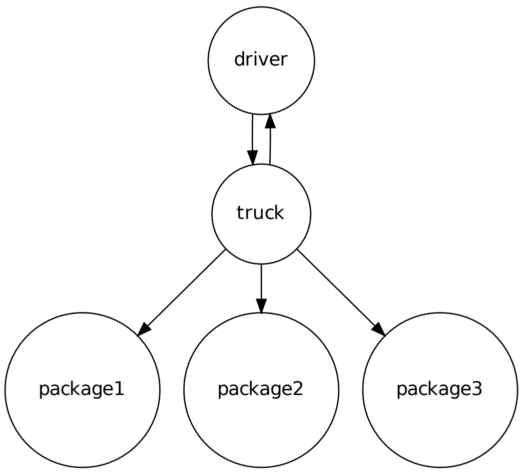
[4]See Appendix D for a descriptions of the DRIVERLOG domain and others referred to in this thesis.


at truck location1
drive truck location1 location2
drive truck location2 location1
at truck location2

(a) The DTG for a truck variable. Nodes represent members of the variables domain.


at driver location1
embark truck driver location1
disembark truck driver location1
driving driver truck1
disembark truck driver location2
embark truck driver location2
at driver location2

(b) The DTG for a driver variable. Nodes represent members of the variables domain.



(c) A causal graph. Arrows indicate a dependency between nodes, such that the outgoing variable may influence the value of the incoming variable. Each node has an associated DTG.

Figure 4.1: A causal graph and a subset of the associated DTGs for a simple logistics problem.

within the relaxed goal-space, but a hypothesis cannot be guaranteed to be non-mutex in the original domain or contain unreachable facts. The latter of these is determined via the grounding process, which must be assumed to be correct, despite often being an overestimation.

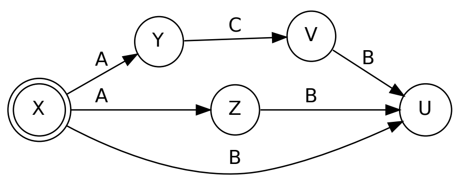
This section is structured as follows. In Section 4.3.1 the goal-space is relaxed to allow a subset of the complete goal-space to be enumerated at runtime, before a method of constructing hypotheses from this goal-space is given in Section 4.3.2. Three suboptimal heuristics are introduced in Section 4.3.3, while the original definition of *work* performed by observations is updated in Section 4.3.5 for a relaxed environment. Section 4.3.6 follows on from this by providing alternatives to the simple *maximum-likelihood* work function, which may be better suited to the relaxed model. The relaxed model features are then rounded off with a discussion of bounded hypotheses, plan-library integration and goal abandonment.

4.3.1 Relaxed Goal-Space

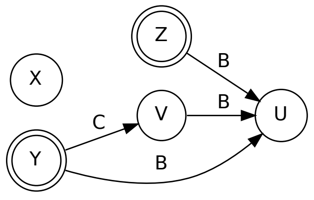
The model presented in the previous chapter reflects many assumptions which are unrealistic in real-life application. This section will present a way of mitigating the inability to detect all mutexes at runtime, but will retain the assumption of an optimal heuristic being available to the observer (which will itself be relaxed later). While it is trivial to detect simple *binary* mutex relations at runtime [23], detecting all mutex relations between goals is as hard as the underlying problem [32]. This forces the assumption of *soundness* in the goal-space to be broken, as there may be goal conjunctions which are invalid due to incomplete mutex information, such as a car being in two locations at once.

Of course, this new goal-space without complete mutex information is itself no more tractable to enumerate than the previous definition of $\mathbb{G}$. Thus, a relaxation is applied in which only *single literal facts* within $\mathbb{G}$ are enumerated, to achieve a relaxed goal-space $\mathcal{G}$. This relaxed goal-space considers only facts which are reachable from the initial state to be goals, reducing the size of the goal-space size from at most $2^{|F|}$ to $|F|$.

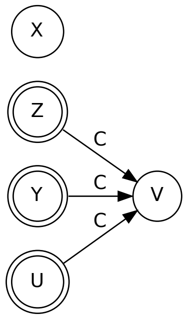
Definition 16. Relaxed Goal-Space
Given a base planning problem with an unknown goal, the corresponding complete goal-space, $\mathbb{G}$, can be relaxed to contain only individual goal literals. This relaxed goal-space, $\mathcal{G} \subseteq \mathbb{G}$, will be equal to an enumeration of single literals in the original goal-space at time t, such that $\mathcal{G} = \pi(F)$.



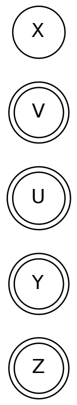
(a) Initial relaxed goal-space



(b) After action A is observed.



(c) After action B is observed.



(d) After action C is observed.

Figure 4.2: The *relaxed* version of the goal-space outlined in Figure 3.3. Only individual literals are enumerated at any time, with any literals which are true in the current state highlighted using double-circles. Actions are still applicable if goal $G \subseteq a_{pre}$.

If left in this form, *completeness* will be lost as any hypothesis will be a single literal, with all members of $\mathcal{G}$ being mutually-exclusive (as was the case in $\mathbb{G}$). However, in practice this can be retained as the next section will show. Figure 4.2 provides an example of this relaxed goal-space when applied to the fully-enumerated goal-space equivalent in Figure 3.3 (page 41). Note that the transitioning behaviour which applied to the previous complete goal-space is retained in the relaxed model.

At this point, the goal-space has been relaxed to allow tractable goal recognition to occur. However, the new single literal relaxation allows for a further change to be made to $\mathcal{G}$, such that it is no longer simply a probability distribution across $\pi(F)$, but rather a probability distribution across multiple *sub-goal-space* distributions.

Extending $\mathcal{G}$ to a Multivariate Form

With the relaxed goal-space, any implementation becomes much closer to previous work on plan and goal recognition, in which there is assumed to be only a single goal that the agent is pursuing [18, 38, 60, 97, 111]. This goal is further assumed to be mutually-exclusive with all other members of the goal-space, meaning the agent ceases executing actions once the goal is achieved for the first time[5].

At first glance, this would also appear to hold true in $\mathcal{G}$, and indeed can without issue. However, the assumption of a single literal goal is not only highly restrictive in practice, but also enforces unnecessary complexity during Bayesian probability updates (Section 3.4.2). This is because most goals in a given problem will *not* be mutually-exclusive with the rest of the goal-space, but rather with only a subset of this. For example, a goal literal representing the location of a package in a logistics domain is unaffected by the value of a driver's location. This is conceptually similar to the use of *variables* in the SAS$^+$ representation outlined above. Each variable $v \in \phi(V)$ will have its current value represented as a member of its domain $D_v \subseteq \pi(F)$, which when viewed in the context of recognition, can itself be considered a probabilistic relaxed goal-space, representing only the most likely final value of D_v. However, instead of applying a distribution across D_v, it is better to make this distribution across each *mutex-set* detected at runtime, $M_{v_1}...M_{v_n}$, for which $D_{v_i} \subseteq M_i$.

Choosing mutex-sets instead of variable domains will allow for more accurate Bayesian inference by including mutex-relations which exist outside that of the variable's domain (as the domain of each variable is itself guaranteed to be mutex). In the context of a pure SAS+ problem, performing recognition across each variable present will result not in goal recognition, but rather *state recognition.* In fact, the use of mutex-sets only partially weakens this statement, and will be explored further in Chapter 5.

If each mutex-set now has an associated probability distribution, a new re-

[5]Here, the term "single literal goal" is defined to be a member of $\pi(F)$. This negates the potential for *high-level goals*, in which a single literal encapsulates the achievement of n members of $\pi(F)$. For instance, in the classic library-based approach to recognition, a high-level goal may encapsulate the execution of an entire plan, but have no mapping to anything within the original set of ground facts. A plan which has the overall goal of `(all_packages_delivered)`, encapsulates the achievement of `(delivered package1)`, `(delivered package2)` etc. but may not exist in the original domain model.

laxed goal-space is effectively formed. Therefore, the overall goal-space can be considered to be a single, relaxed distribution across $n \in [1 : |\pi(F)|]$ relaxed goal-spaces, each of which is a further distribution across its own members. This high-level distribution shall be referred to as the *multivariate* relaxed goal-space $\mathcal{G}^V$, and is illustrated in Figure 4.3[6].

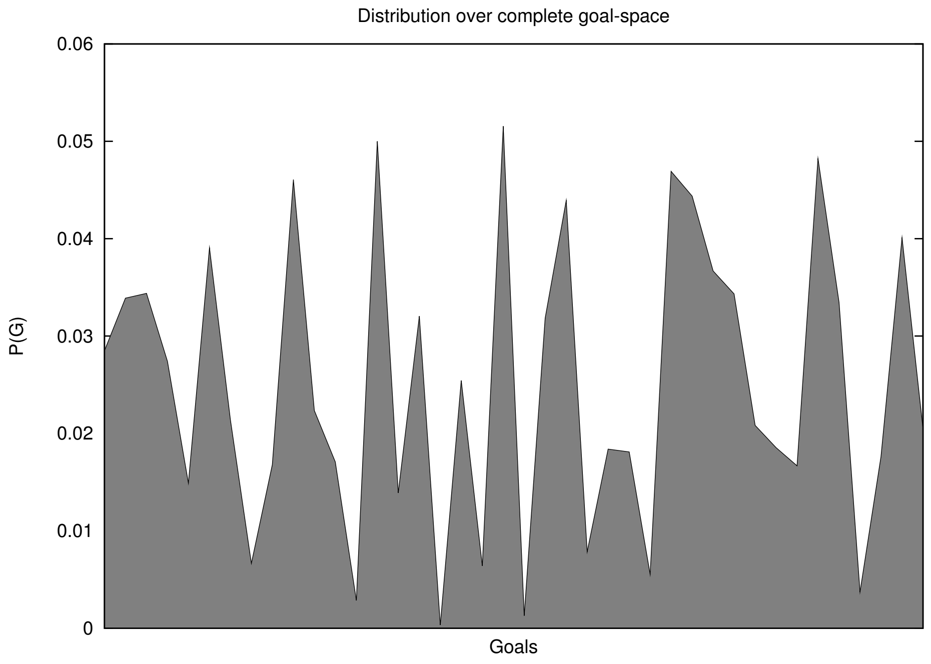
Definition 17. Multivariate Goal-Space
Given a series of *mutex-sets* $M_1...M_n$, a relaxed goal-space can be constructed for each, such that $M_i \rightarrow \mathcal{G}_i$. A *multivariate relaxed goal-space* $\mathcal{G}^V$ is a probabilistic distribution across these n *sub-goal-spaces.*

To place this in the context of heuristic-based goal recognition, consider the following example in which a package, `p1`, must be moved from location `s1` to `s2`. In the original goal-space $\mathbb{G}$, the heuristic estimate of `(at s2 p1)` would be computed, along with all other members of $\mathbb{G}$ in order to determine the amount of work $W_{ML}(G)$ the agent's plan has expended on achieving G. This would then be used in computing the posterior probability of $G =$ `(at s2 p1)` being the true goal. As $\mathbb{G}$ was complete, the denominator of Bayes' theorem would include the likelihood probabilities of all goals, producing an extremely small posterior probability.

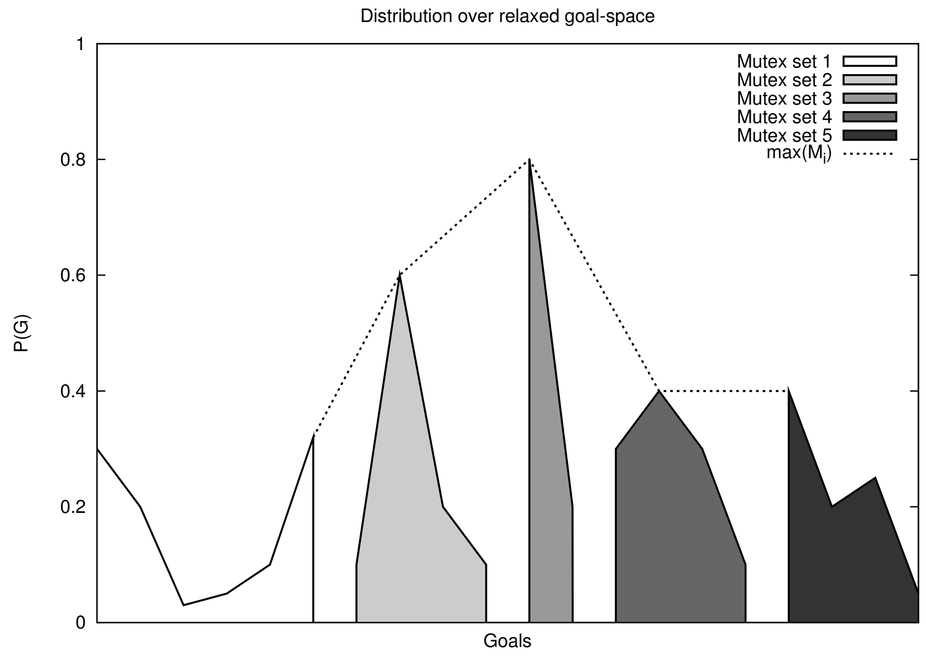
In the multivariate model, only members of the associated mutex-set are used in the Bayesian denominator (in this case goals such as `(at s1 p1)`, `(at s1 p3)` and `(in p1 truck)`), resulting in much larger posteriors with fewer calculations required. However, there is one further advantage to this representation — the ability to retain *completeness* in the relaxed goal-space. Given that each sub-goal-space represents a set of mutually-exclusive facts, all hypotheses present in the original goal-space $\mathbb{G}$ can still be produced as a conjunction of k literals taken from the n mutex-sets, where $1 \leq k \leq n$. Note that $\mathcal{G}^V$ is still referred to simply as "the goal-space", and that "all goals in $\mathcal{G}^V$" is the union of all members of the sub-goal-spaces, $\bigcup_{i=1}^{|\mathcal{G}^V|} \mathcal{G}_i \in \mathcal{G}^V = \pi(F) = \phi(F)$, with the exception of any SAS$^+$-specific facts generated during translation.

Here "completeness" is, in fact, potentially an overestimation of the original goal-space. This is once again due to the inability to detect all mutexes at runtime [32]. As such, some mutex-sets may be incomplete, so while any hypothesis

[6]The term "multivariate" is somewhat misleading as it implies that each sub-goal-space is indeed a distribution across each variables domain and not a mutex-set as specified. However, it serves as a useful associative term in aiding understanding of the relaxed model.



(a) A sample probability distribution across a complete goal-space, $\mathbb{G}$.



(b) A different problem, demonstrating the probability distribution across mutex-sets, $M_1 ... M_5$, in the relaxed multivariate goal-space, $\mathcal{G}^V$. Individual mutex sets are highlighted, and can be considered to be discrete relaxed goal-spaces. Note that overlapping goal-spaces have been omitted for simplicity.

Figure 4.3: The difference in probability distributions between the original complete goal-space and a separate multivariate relaxed goal-space.

produced from the multivariate relaxed goal-space is valid in terms of the known mutexes, these may not necessarily map directly to the full set of undetectable mutexes. The construction of hypotheses will be discussed in further detail in Section 4.3.2.

Including *All False* Goals in $\mathcal{G}_i$

Thus far, the model presented has had the implicit assumption that all goals must exist as *positive* literals within $\mathcal{G}$. In the complete model this assumption translates to only considering the achievement of a literal as a goal, rather than the negation. However, in practice observations can also be used to indicate what is *not* a goal, by considering those goals which have consistently had their heuristic estimate increase. These *all false* goals are not the individual negation of literals (such as `(not(a))` w.r.t `a`), but rather the negation of *all* mutex goals. That is, if facts A and B are mutex then there may be an implicit case where neither are true.

In the original complete goal-space, considering situations where all members of a mutex-set are false results in an extra literal being added, $|\mathbb{G}| = 2^F + 1$. This results in the *all false* goal (denoted $G_\emptyset$) being considered a valid hypothesis. However, in practice this is extremely unlikely behaviour, as it translates to every goal in the goal-space becoming further away from its original heuristic estimate. This itself is only possible if every action in the problem has empty add effects.

In the relaxed multivariate model the potential of these *all false* goals being the true goal is a far more likely consideration. Each sub-goal-space $\mathcal{G}_i = \{G_1...G_n\}$ therefore has a respective *all false* goal added to it, $\mathcal{G}_i = \{G_1...G_n, G_\emptyset\}$. Each new goal $G_\emptyset$ is itself treated as if it were a positive literal, in that the work function for it remains as in Equation 3.2, where work is considered to be the ratio of observations which have been helpful. However, in this special case, the definition of what "nearer" means must be inverted in order that an observation is considered helpful for $G_\emptyset$ only if all mutex goals have become further away. For example, if an observation causes goals A, B and C to have their estimates increased (where $\mathbb{G} = \{A, B, C\}$), then *no* work has been put towards their achievement. In fact, the action has served only to make it harder to achieve any of these goals, therefore making it helpful towards achieving $G_\emptyset$. This behaviour is shown in equation 4.1, where G^+ is a standard goal (i.e. not an *all false* goal) in goal-space $\mathbb{G}$.

$$W_{ML}(G_\emptyset|O_t) = \frac{\sum_{i=1}^{t} \begin{cases} 1 & \text{if } h_i(G^+) < h_{i-1}(G^+) \text{ and } i > 0, \forall G^+ \in \mathbb{G}, \\ 0 & \text{otherwise} \end{cases}}{|O|} \tag{4.1}$$

Were the above behaviour to occur in the original complete goal-space, the only goal remaining would be $G_\emptyset$, as all others could no longer be achieved with an optimal plan. In the relaxed model, this rather indicates that the values held within the second sub-goal-space are not relevant to the agent, and that they may be side-effects of the plan for achieving the true goal.

In reality, mutex-sets virtually never contain facts which can be completely negated under certain circumstances. However, there is one exception to this which *all false* are always required — the situation in which the sub-goal-space is of size $|\mathcal{G}_i| = 1$. Here, the *positive* goal, G^+, is mutex only with its negation, such as an image having been transmitted or not in the Rovers domain[7]. This fact is independent of all others in the goal-space, which would result in a Bayesian posterior of $\mathrm{P}(G|O) = 1$ for all observations. Under these circumstances, the inclusion of $G_\emptyset$ in $\mathcal{G}_i$ allows for the correct posteriors to be computed, and prevents G^+ from always being put forward as a valid goal candidate[8].

While the multivariate goal-space $\mathcal{G}^V$ may contain several sub-goal-spaces, by itself it represents the distribution across the union of all goals in these sub-goal-spaces $\mathcal{G}^V = \bigcup_{i=1}^{|\phi(M)|} \mathcal{G}_i$. Now recall that in the complete goal-space it is trivial to extract a hypothesis, as it is simply the member of $\mathbb{G}$ with the highest probability. However, in the relaxed goal-space $\mathcal{G}^V$, only single literal goals can be considered using this approach. Given that it is unlikely the agent's goal will be only a single literal, a method for extracting *conjunctive* hypotheses from the relaxed goal-space is necessary.

4.3.2 Conjunctive Hypothesis Extraction

In the multivariate model, each sub-goal-space will have its own distribution across its member goals. The goal which has the highest associated probability is therefore the natural candidate for any hypothesis produced. As there are n

[7]See Appendix D.

[8]In domains such as Rovers, sub-goal-spaces which would otherwise be of size 1 are numerous. If the negation of the positive literals contained within these spaces is not considered, hypotheses can become extremely large.

sub-goal-spaces in $\mathcal{G}^V$, hypotheses can be constructed by taking each of these maximum-probability goals from each sub-goal-space. This is referred to as a *relaxed hypothesis.* That is, the relaxed hypothesis formed will comprise of the set of goals from each sub-goal-space which have the maximum probability.

Definition 18. Relaxed Hypothesis
Given a multivariate relaxed goal-space $\mathcal{G}^V$, a relaxed hypothesis, Hyp is defined to be the union of maximal members of each sub-goal-space $\mathcal{G}_i \in \mathcal{G}^V$, such that $Hyp = \bigcup_{i=1}^{|\mathcal{G}^V|} [\arg\max \mathrm{P}(G)], \forall G \in \mathcal{G}_i$.

Note that there may be situations in which a goal appears in more than one mutex-set, and thus more than one relaxed goal-space in $\mathcal{G}^V$. This is *not* the manifestation of the loss of soundness caused by moving to a relaxed model, but rather an example of the natural overlap between members of $\mathcal{G}^V$. Without taking action against this, unreachable hypotheses can potentially be created. For example, if $\mathcal{G}_1 = \{A, B\}$ and $\mathcal{G}_2 = \{B, C\}$, and B has the maximum probability in both distributions, the relaxed hypothesis produced would simply be $Hyp = B$, which is both reachable and minimal in size. However, if instead A has the maximum probability in $\mathcal{G}_1$, the hypothesis would be $Hyp = \{A, B\}$.

IGRAPH mitigates the problem of two goals being mutually-exclusive in a hypothesis by filtering the set of goals chosen to obtain a *greedy relaxed hypothesis.* This is a sound hypothesis in the relaxed goal-space, in that it will be free of all known mutually-exclusive goals.

A naïve approach to this would be to always retain the goal which has the higher probability between two mutex goals. However, this is flawed and can lead to a *biased* hypothesis which always favours certain goals. Specifically, goals which appear in multiple sub-goal-spaces will have varying probabilities across these spaces which may vary greatly. For example, a goal may have probability $\mathrm{P}(G) = 0.8$ in $\mathcal{G}_1$, but probability $\mathrm{P}(G) = 0.05$ in $\mathcal{G}_2$. In the former case G will always be selected for the relaxed hypothesis, but there may exist a goal in $\mathcal{G}_2$ which has a higher probability than G (in this sub-goal-space), resulting in a mutex hypothesis being formed. If G is simply retained because it has a higher probability in $\mathcal{G}_1$, then it may bias the hypothesis toward goals which have, in reality, not been supported by the observed plan.

This behaviour is caused by sub-goal-spaces whose size is small (e.g. $|\mathcal{G}_i| = 2$), as the "helpfulness" of observations are distributed across all members of $\mathcal{G}_i$. Therefore smaller goal-spaces which feature a goal that has become heuristically

Algorithm 3 This algorithm greedily extracts a hypothesis from the set of maximal goals in each sub-goal-space as given in Definition 18. Before the maximum probability goal from each sub-goal-space is added to the hypothesis, it is checked against all current members of the hypothesis to determine if it is mutually-exclusive with them. If it is, the fact which is retained is determined by the *doTieBreak* function, given in Algorithm 4. Finally, any *strictly terminal* facts which have been achieved during observation are added to every hypothesis (see Definition 23).

```
Require: $getHypothesis()$
1:  $Hyp := max(\mathcal{G}^V)$ {Set of highest-probability goals}
2:  $queue := \forall G \in Hyp$
3:  $inferior := \{\}$
4:  while $!isEmpty(queue)$ do
5:    $G_{high} := pop(queue)$
6:    if $G_{high} \in inferior$ then
7:      continue
8:    end if
9:    for all $G_Q \in queue$ do
10:     {Find if goals are mutex}
11:     if $mutex(G_{high}, G_Q, \mathcal{G}^V)$ then
12:       {If so, find if $G_{high}$ is preferred, where $G_{high}, G_Q \in \mathcal{G}_i$ and $\mathcal{G}_i \in \mathcal{G}^V$}
13:       if $doTieBreak(G_{high}, G_Q, \mathcal{G}_i)$ then
14:         $add(inferior, G_Q)$
15:       else
16:         continue
17:       end if
18:     end if
19:   end for
20: end while
21: $Hyp^G := \{Hyp \setminus inferior\}$
22: $Hyp^G := Hyp^G \cup ST_{achieved}$
23: return $Hyp^G$
```

closer will receive higher posteriors than those which have more members. It follows that the probability alone is not sufficient for comparing goal candidates across multiple goal-spaces.

Should the above situation occur, one of the mutex goals must be removed from the hypothesis. As probabilities are inapplicable, a series of tie-breaks are performed to determine which goal is the most likely candidate. Algorithm 3 shows this greedy filtering process, the output of which is a valid hypothesis in the relaxed goal-space.

This algorithm considers first which goal has had the most work put towards

Algorithm 4 A tie-breaking algorithm for determining whether goal G_A should be given precedence over mutex goal G_B. The algorithm takes in two goals G_A and G_B, and an optional parameter $\mathcal{G}_i$, corresponding to the sub-goal-space both goals belong to (if applicable). In the case of non-mutex goals which both appear in $\mathcal{G}_i$, the algorithm simply returns which has the higher associated probability. However, if these do not both belong to $\mathcal{G}_i$, various tie-breaking criteria are checked. These are, in order; the total distance moved towards the goal ($\Delta^+ h(G)$, more movement preferred); the layer the goal exists on the causal-graph ($CG_L(G)$, higher layers preferred — this concept will be introduced in Section 4.4.3) and finally the distance remaining to the goal ($h(G)$, further goals preferred). If none of these criteria are met, a random coin-flip is used.

```
Require: $doTieBreak(G_A, G_B, \mathcal{G}_i)$
 1: if $!mutex(G_A, G_B, \mathcal{G}_i)$ and $\mathrm{P}_{\mathcal{G}_i}(G_A) > \mathrm{P}_{\mathcal{G}_i}(G_B)$ then
 2:   return true
 3: else if $!mutex(G_A, G_B, \mathcal{G}_i)$ and $\mathrm{P}_{\mathcal{G}_i}(G_A) < \mathrm{P}_{\mathcal{G}_i}(G_B)$ then
 4:   return false
 5: else
 6:   if $\Delta^+ h(G_A) > \Delta^+ h(G_B)$ then
 7:     return true
 8:   else if $\Delta^+ h(G_A) < \Delta^+ h(G_B)$ then
 9:     return false
10:   else
11:     if $CG_L(G_A) > CG_L(G_B)$ then
12:       return true
13:     else if $CG_L(G_A) < CG_L(G_B)$ then
14:       return false
15:     else
16:       if $h(G_A) > h(G_B)$ then
17:         return true
18:       else if $h(G_A) < h(G_B)$ then
19:         return false
20:       else
21:         if $rand() < 0.5$ then
22:           return true
23:         else
24:           return false
25:         end if
26:       end if
27:     end if
28:   end if
29: end if
```

its achievement, as this value informs the probability of it being the goal. After this, the *layer* at which each goal appears in the causal graph is analysed. A full description of goal layers shall be enumerated in section 4.4.3. Briefly, goals which appear at lower layers of the causal graph are often more likely to appear as goals in a planning problem as they are influenced by other goals, but exert no influence themselves. For example, packages in DRIVERLOG exist only to be moved to other locations — their location does not influence the movement of trucks or drivers.

Should the causal graph layer be equal for both goals, the distance remaining to the goal is considered, with further goals being preferred. Finally, a random coin-flip is used to determine which goal will be retained.

The exceptions to this process are *all false* goals. If an *all false* goal is selected as being the most probable member of a sub-goal-space, it is simply omitted from the hypothesis prior to tie-breaking.

Final Hypotheses

If an agent is known to have finished their plan, then their goal must exist within the final state, $G^* \in S^*$. Therefore, the above hypothesis extraction method is no longer relevant, as the probability associated with each fact in the goal-space is of no use. That is, having the highest probability in the associated sub-goal-space does not matter, if the fact itself is not true in S^*. To this end, given that it is known there will be no more observations, the system should be able to determine which of the goals in S^* are the true goal and not simply side-effects[9].

The problem of extracting a goal given the complete plan has previously been tackled by Hong [89], in which a *plan graph* [23] is constructed to determine those facts in the final state which have the most actions contributing towards their achievement. This is done by tracking the causal-links between actions from the initial to final state. While this approach is valid for IGRAPH, it would require enumerating 2^k fact combinations, where $k = |S^*|$ in order to determine which subsets have had the most work contributed towards them. If instead only single literals are considered, there is a risk that goals which are achieved using relatively few observations will be ignored as irrelevant. Therefore, IGRAPH adopts a different approach to final hypothesis construction.

The problem of extracting goals from the final state is common to both the

[9]Given that the world is fully-observable and deterministic, any final hypothesis is guaranteed to be *sound* in the original state and goal-spaces (unlike intermediate hypotheses which are *relaxed*).

complete and relaxed goal-space models. In both cases, regardless of whether all conjunctive goals or only single literals are enumerated, any combination of facts present in S^* will necessarily exist in $\mathbb{G}$. It is therefore impossible to determine which subset of the final state is actually the agent's goal without placing some assumptions on the model. To this end, final hypotheses in IGRAPH are based upon selecting those which have the highest *stability*, as defined below. Using Hong's approach of selecting the goal which has had the most heuristic movement towards it can also be viewed as a valid technique [89], but is rejected as it does not take movement of resources into account. For example, a truck with limited capacity delivering multiple packages to a destination will need to make the same journey several times, leading to a higher value for total heuristic movement than the packages themselves (which are most likely the true goal).

Definition 19. Goal Stability
Given a fact/goal, $G \in \mathbb{G}$, the associated *stability* of that goal, $\Upsilon(G) \in (0 : 1]$, is the ratio of times the goal has had its value changed since its first achievement in the plan. Each fact has a value of $\Upsilon(G) = 1$ until its first achievement, after which the previous statement applies.

$$\Upsilon(G) = \begin{cases} 1 & \text{if } G \text{ has been achieved 0 or 1 times,} \\ \dfrac{1}{switches(G)} & \text{otherwise} \end{cases} \tag{4.2}$$

Equation 4.2 provides a further expansion of this, where $switches(G)$ is the number of times the goal has been deleted or re-achieved since its first achievement. For example, if goal G is first achieved on observation 10, then deleted and re-achieved in observations 11 and 12 respectively, its value has changed twice — once to *false* after the first achievement (which is not counted) and then to *true*. This results in a value of $\Upsilon(G) = \frac{1}{2}$, which would remain until G was again negated. Note that *all false* goals will always have a stability of 1, as by-nature they cannot be negated.

IGRAPH applies the assumption that an agent will always strive to maintain a goal literal as true once it is first achieved. Computing the *stability* of a goal therefore provides a means of ranking goals, something which can be applied to final hypothesis construction.

Definition 20. Final Hypothesis
Given a final state S^*, the *final hypothesis* $Hyp_\Omega \subseteq S^*$, is equal to all facts true in this state $f \in S^*$ for which $\Upsilon(f) \geq \eta$, where $\eta \in [0:1]$.

The final hypothesis is therefore all facts in the final state which have a stability of at least η. A value of $\eta = 1$, translates to the assumption that the agent will always keep a goal literal true, while the opposite extreme of $\eta = 0$ states that any fact true in the final state can be considered a goal.

Note that it is not considered valid for the final hypothesis to be empty, $Hyp_\Omega = \emptyset$. While this is valid behaviour in *intermediate* hypotheses, it is only possible as a side-effect of undetectable mutexes and is still considered "unexpected" behaviour — the assumption of the agent having at least one positive literal goal is retained, and as the final state must contain the goal, it therefore follows that $G^* \in S^*$.

This section has described a means of relaxing the original goal-space while still being able to construct conjunctive hypotheses, in order that its application becomes tractable. However, the assumption of an optimal heuristic being available to the observer has been retained, despite also being intractable to produce in a domain-independent context [32]. The next section will describe several *sub-optimal* domain-independent heuristics which have been previously been applied in the field of planning, in order that these can be evaluated in IGRAPH as a means of approximating an optimal heuristic.

4.3.3 Relaxed Heuristics

The approximation of optimal, domain-independent heuristics is a topic of great interest within the planning community [17, 81]. Yet, suboptimal heuristics have been shown to be highly informative in plan generation [27, 78, 79, 88] to the point where classical planning can be performed on extremely large and complex problems. However, while planning is commonly moving towards tackling more complex problem representations, this work and that of Ramirez and Geffner [145, 146, 147] represents the first time such heuristics have been applied in recognition. Naturally, this is because a plan-library is commonly available, thus negating the need for heuristics. This is unfortunate, as it limits the ability to perform recognition on any problem without considerable investment in library construction.

IGRAPH utilises suboptimal planning heuristics which have shown to be informative across multiple planning domains, in order that goal recognition can be performed without prior knowledge of the domain or executing agent. A heuristic is suboptimal if it can — in addition to being accurate — over or under-estimate the distance to goal G. For example, a heuristic may produce a value of $h_t(G) = 14$ at time t, and a value of $h_{t+1}(G) = 15$ after the next observation, despite the action contributing towards achieving G. This risk can be somewhat mitigated by using an *admissible* heuristic, which guarantees to never over-estimate the distance to G, where $h(G) \leq h^+(G)$, $\forall G \in \mathbb{G}$.

IGRAPH implements three heuristics which represent milestones in the development of domain-independent planning. The first, the *max* heuristic (h_{max}) [27] and the HSP planner which introduced it [26], demonstrated that both forward and backward-chaining planning was possible using a domain-independent heuristic. However, h_{max} is extremely uninformative by modern standards (and was regarded as poor even in 2000), but is the only heuristic implemented in IGRAPH which is admissible. It therefore serves as a useful baseline for other heuristics to be judged against.

The second heuristic implemented is the FF heuristic, h_{ff} [88]. This quickly subsumed h_{max} (and associated h_{add} heuristic [26] detailed below) as the state-of-the-art, and the associated planner was successful in several International Planning Competitions (IPC) [47, 117]. Indeed, the heuristic itself has also been a component of multiple successful planners [39, 43, 79, 83, 85, 86, 163].

Finally, unlike the previous two heuristics, the *context-enhanced additive* heuristic h_{cea} [80], uses a SAS$^+$ based representation [10] to search for relaxed paths through the associated causal graph and DTGs. It is more accurate than h_{ff} on some problems, such as those which demonstrate a high branching-factor, but can sometimes overestimate the distance to goals which are made up of several literals.

Each of these heuristics will now be presented in more detail. Note that only a single heuristic is used during recognition.

The Max Heuristic

The *max heuristic*, denoted as h_{max}, is one of the oldest domain independent planning heuristics. Proposed by Bonet and Geffner in 1997 [27], it signalled that planning was tractable on multiple unseen problems using a single common heuristic. h_{max} offers an interesting baseline heuristic for application in a goal recognition system, as it is both fast to compute and *admissible*. However, it is also regarded as uninformative, in that it will usually heavily underestimate the work remaining, which can lead to poor values being computed, particularly in domains which feature actions with a large set of preconditions.

h_{max} is built upon the *delete-list* relaxation [28], in which the delete-effects of actions are ignored during calculation of the heuristic cost to a goal. As its name indicates, the cost of a conjunctive goal G is the maximum cost of achieving each of the individual literals. This is computed recursively from goal G, until $\forall g \in G, g \in S$, where S is the current state. Figure 4.4 shows this process for a simple problem. This is done using the following formula, where $A_{achievers}(g)$ is the set of actions for which $g \in a_{add}$.

$$h_{max}(G) = \max_{g \in G} \begin{cases} 0 & \text{if } g \in S, \\ \min\limits_{a \in A_{achievers}(g)} [1 + h_{max}(a_{pre})] & \text{otherwise} \end{cases} \tag{4.3}$$

Table 4.1 shows heuristic estimates from the initial state to the goal, using h_{max} across 20 problems in the Rovers domain, taken from IPC5. The second column indicates the value of h_{max}, while the fourth indicates the shortest solution length of the best planner in the competition. As the table clearly demonstrates, for non-trivial problems h_{max} provides extremely poor estimates. It is included as a means of demonstrating whether heuristic-based goal recognition is possible even when the heuristic itself is uninformed, and as a baseline for comparing against other heuristics.

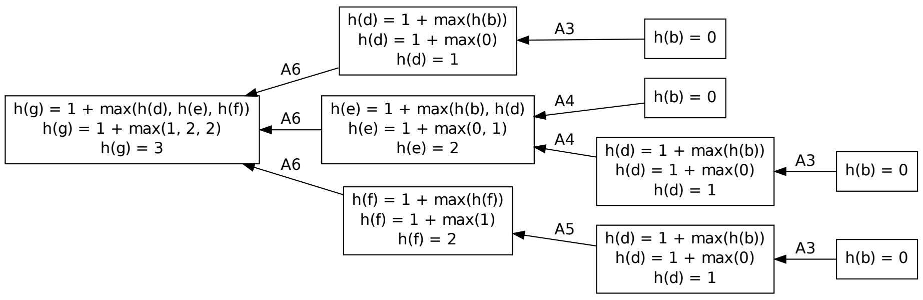


Figure 4.4: The computation of the cost for goal g using the h_{max} heuristic.

Problem	h_{max}	h_{ff}	$\lvert P\rvert$	**Problem**	h_{max}	h_{ff}	$\lvert P\rvert$
1	4	10	10	**11**	5	30	32
2	3	9	8	**12**	4	23	19
3	4	11	11	**13**	4	42	46
4	3	10	8	**14**	4	31	31
5	4	20	22	**15**	3	37	42
6	4	30	36	**16**	3	36	41
7	3	15	18	**17**	4	42	50
8	4	28	26	**18**	4	43	46
9	4	27	31	**19**	4	70	69
10	3	33	35	**20**	4	83	99

Table 4.1: Heuristic estimates for the h_{max} and h_{ff} heuristic, versus the length of the best solution found at IPC5 on the Rovers domain.

The FF Heuristic

One of the best known heuristics in the planning community is the *Fast Forward* (FF) heuristic. This and the eponymous planner which first implemented it [88] quickly became the baseline against which all other planning research was judged [47, 117]. The FF heuristic is both fast and informative, and further can be computed in polynomial time [82], although computation of an optimal cost even in the relaxed problem is still NP-hard [32].

Like h_{max}, the problem is relaxed to remove all delete-effects from actions and mutexes are ignored. The cost of a goal G is determined by the FF heuristic to be the length of the *relaxed plan* $\mathcal{P}$ which achieves it, where a relaxed plan is simply a valid plan within the relaxed state-space. This relaxed state-space is represented by constructing a *relaxed planning graph* (RPG) — a graph of alternating *fact* and *action* layers, as shown in Figure 4.5.

During RPG construction, the graph is initialised with a single fact layer, F_0, which is equal to the facts present in the initial state. An iterative process then begins in which two further layers are created per iteration, with both layers classed as layer k. The first is an action layer, A_k, containing all actions for which $a_{pre} \subseteq F_{k-1}$, while the second is a fact layer, F_k, equal to $F_{k-1} \cup a_{add}, \forall a \in A_k$. All facts in the previous layer are guaranteed to be added through the automatic addition of *no-ops* — stub actions with no preconditions or effects which map to every fact present in the previous layer. This process then repeats until $G \subseteq F_n$. Alternatively, if no goal is specified construction stops once $F_n = F_{n-1}$ (with the duplicate layer F_n ignored), in which case the RPG is said to have *stabilised.* The relaxed plan is then extracted, starting at the first fact layer which contains all literals in G. It is likely that the relaxations imposed during RPG construction will cause the relaxed plan to be *invalid* in the original state-space, such that $\mathcal{P} \notin \mathbb{P}$, hence it cannot simply be used as the final plan.

The estimate itself is constructed using the *additive heuristic* [27] (see Equation 4.5). However, unlike the original application of this, h_{ff} computes the cost from the RPG. This allows positive interactions to be included in the relaxed plan, which lowers the final cost. For example, Figure 4.5 shows how h_{ff} returns a value of $h(g) = 4$, while h_{add} returns a cost of $h(g) = 6$ on the same problem, as shown in Figure 4.6.

To estimate the cost of achieving G, a set of *open goals* is created which initially contains all single literals $g \in G$. A relaxed plan is then regressively extracted from the RPG, starting at the first layer for which $G \subseteq F_n$. For each literal g, the first action in the set of actions that achieves g and appears in action layer A_n is chosen to be added to the relaxed plan. If this set contains a *no-op*, this will always be chosen over normal actions. The effects of selected actions are removed from the set of open goals and the union of all preconditions added. This process iterates until the set of unachieved goals is a subset of the initial state.

Table 4.1 (page 86) shows the improvements in heuristic accuracy if h_{ff} is used in the ROVERS domain, versus the h_{max} heuristic. h_{ff} shows a marked improvement in estimating the final solution length, although it should be noted that it is possible that some of the best solutions found were *produced*, at least in part, using the h_{ff} heuristic.

$$h_{ff}(G) = |\mathcal{P}| \tag{4.4}$$

$$h_{add}(G) = \sum_{g \in G} \begin{cases} 0 & \text{if } g \in S, \\ \min\limits_{a \in A_{achievers}(g)} [1 + h_{add}(a_{pre})] & \text{otherwise} \end{cases} \tag{4.5}$$

The Context-Enhanced Additive Heuristic

Unlike the previous two heuristics, the *context-enhanced additive heuristic*, h_{cea}, is not formulated for propositional-logic (as represented by PDDL). Instead, the SAS$^+$ formalism as described in Section 4.2.1 is used [11]. Using the work of

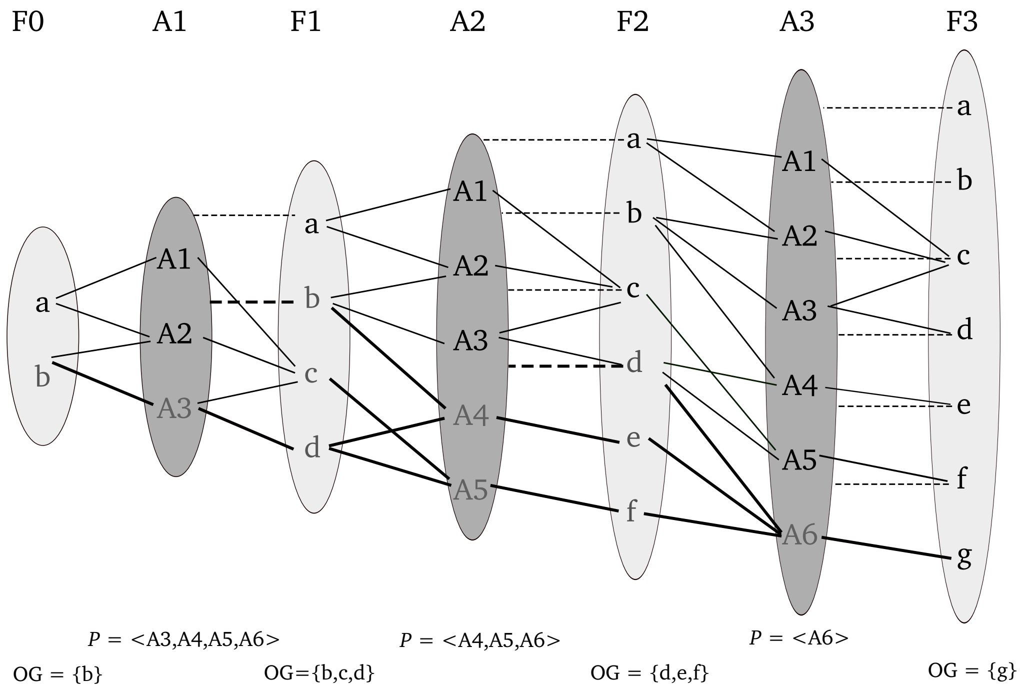


Figure 4.5: A relaxed planning graph showing one possible variation of relaxed plan extraction for the single literal goal g in layer F_3 using the FF heuristic, with a final cost of $h(g) = 4$. OG is the set of *open goals* which should be satisfied by the previous action layer and $\mathcal{P}$ is the current relaxed plan. Note that *no-ops* are shown by dashed links and are always preferred to actions.

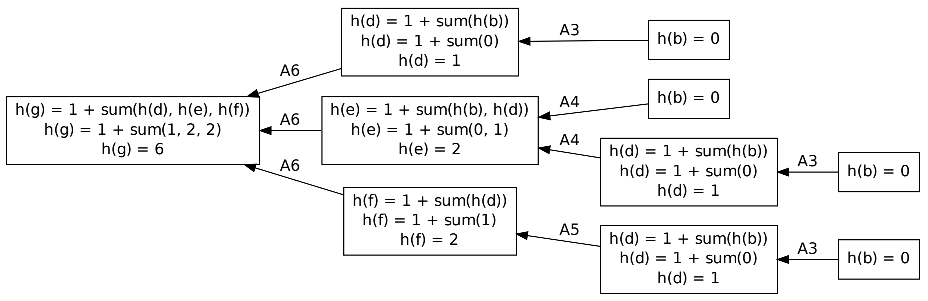


Figure 4.6: The computation of the cost for single literal goal g using the *additive heuristic*. The cost of a goal is the sum of achieving the goal itself plus the cost of achieving the preconditions of the achieving action. Positive interactions between actions are not included in the final value, which leads to over-estimation of the true cost. For example, the cost of achieving facts b and d are both included twice.

Helmert [78], PDDL problems can be translated into an equivalent SAS$^+$ representation, which allows the h_{cea} heuristic to be applied.

However, before h_{cea} can be described, the problem setup and prior work must be given. This has partly been covered through detailing the h_{add} heuristic, but also requires that the *causal graph* heuristic be expanded upon, as these form the basis of h_{cea}.

Causal Graph Heuristic The causal graph heuristic [78] uses the *causal graph* and *domain transition graphs* generated by Helmert's translator, to compute a path from the current state to the goal. This path is in fact the conjunction of other paths through various SAS$^+$ *subproblems* generated during search as follows.

Every variable, $v \in \phi(V)$, in the causal graph has a corresponding domain transition graph, DTG(v). An edge $\langle a, b \rangle$ in the causal graph indicates that at least one transition in DTG(b) has a precondition that the value of DTG(a) be set to a specific value. In this situation, b is referred to as the *high-level variable*, v_H, and each object connected by an incoming edge is referred to as a *low-level variable*, v_L. If v_H has no outgoing edges (that is, it is a *leaf* in the graph), and all low-level variables connected via an incoming edge are *roots* of the graph, then the problem is a SAS$^+$-1 problem.

Definition 21. SAS$^+$-1 Problem
A SAS$^+$-1 problem, Φ, is a SAS$^+$ problem, ϕ, in which the causal graph contains a single high-level variable v_H and all other variables are low-level variables of v_H. The single goal, G^*, of Φ is for v_H to transition from its current value to $G^* \in D(v_H)$, where $D(v_H)$ is the domain of the high-level variable.

For a given SAS$^+$-1 problem and goal G^*, a *plan* is any sequence of actions which enables the high-level variable to transition to G^*. This is achieved through applying Dijkstra's algorithm to find the shortest path to all nodes in DTG($\mathrm{v_H}$) from the current value $s_0(v(H))$, as shown in Algorithm 5. As with standard Dijkstra application, the shortest known plan in the queue is expanded first.

The solution output by the *solveProblem* algorithm is an optimal plan for the achievement of the single literal goal G^*, as the low-level variables do not rely on any other variables in the causal graph.

This principle can be applied to the original SAS$^+$ problem by considering each goal $g \in G^*$ to be a separate SAS$^+$-1 problem, as shown in Algorithm 6. However, before this can take place, any cycles within the causal graph must be

Algorithm 5 solveSubproblem

```
$G^* := v_H \in D_v$
$plans(d_H) := \emptyset, \forall d' \in D_H$
$plans(s_0(v_H)) := \langle\rangle$
$queue := D_H$
while $queue \neq \emptyset$ do
    $d_H := \min(queue)$
    $\mathcal{P} := plans(d_H)$
    $S := plan(s_0)d_H))$
    if $d_H = G^*$ then
        return $\mathcal{P}$
    else if $plans(d^i_H) = \emptyset, \forall d^i_H \in queue,$ then
        Failure — no plan possible
    end if
    for all $\langle d, d'\rangle \in DTG(d_H)$ do
        $\mathcal{P}_L := \langle\rangle$
        for all $v_L \in (a_{pre} \setminus d_H)$ do
            $\mathcal{P}_L := \mathcal{P}_L + dijkstra(DTG(v_L), S(DTG(v_L)), a_{pre}(DTG(v_L))$
        end for
        $\mathcal{P}' := \mathcal{P} + \mathcal{P}_L + \langle d, d'\rangle$
        if $|\mathcal{P}'| \leq |\mathcal{P}|$ then
            $plans(d_H) := \mathcal{P}'$
        end if
    end for
end while
```

broken, in order that $\forall\langle v, v'\rangle \in CG_{edges}, \nexists\langle v', v\rangle \in CG_{edges}$. By doing this, the algorithm is guaranteed to terminate, but that information about the problem may also be lost. The decision as to which edge is removed ($\langle v, v'\rangle$ or $\langle v', v\rangle$), is based upon whether v or v' appears in fewer action preconditions. The variable which has the least occurrences has its outgoing edge removed from the causal graph, thus respecting as many preconditions as possible.

With an acyclic graph, the heuristic estimate to a conjunctive goal G is the sum of each individual goal's estimate, as given in Equation 4.6 and Algorithm 6.

$$h_{cg}(G) = \sum_{i=1}^{|G|} h_{cg}(g_i) \tag{4.6}$$

Incorporating Cyclic Causal Graphs The acyclic graph required for h_{cg} to terminate is problematic, as many planning problems contain cyclic dependencies.

Algorithm 6 solveProblem

```
$\bar{\mathcal{P}} := \emptyset$
for all $g \in G^*$ do
    $\mathcal{P}_g := solveSubproblem(g)$
    $\bar{\mathcal{P}} := \bar{\mathcal{P}} + \mathcal{P}_g$
end for
return $\bar{\mathcal{P}}$
```

When such cycles are broken, the result of the Dijkstra search can be extremely poor heuristic estimation in domains with combinatorial aspects (such as block stacking). Helmert and Geffner address this problem by combining the *additive* heuristic with h_{cg} to derive the context-enhanced additive heuristic, h_{cea} [80].

In the above description of h_{cg}, SAS$^+$-1 problems are trivial to solve, as there is only one root/high-level variable, and each connected leaf/low-level variable has no further children. In this case, the plan to solve the goal is optimal. Helmert and Geffner note that this is also the case if the additive heuristic is applied, and derive h_{cea} as a method of obtaining heuristic estimates when the goal is *not* a simple SAS$^+$-1 problem. Equations 4.7 and 4.8 show how h_{add} is transformed when applied to a SAS$^+$ problem, where S is a state (initialised to the initial state) and G is a set of variable-value pairs indicating the goal, which can be considered a partial state. The notation x_s indicates that variable x holds the value of variable x in state S. By maintaining a representation of the current state as execution proceeds, h_{cea} can provide more accurate estimates than h_{add}, which only uses the initial state to determine whether goals have been met (hence the term "context-enhanced additive heuristic"). Note that if h_{cg} is computable, it will be equal to h_{cea}.

$$h_{add}(S) = \sum_{x_g \in G} h_{add}(x_g | x_S) \tag{4.7}$$

$$h_{add}(x|x') = \sum \begin{cases} 0 & \text{if } x = x', \\ \min\limits_{o:P \to x} c(o) + \sum\limits_{y \in P} h_{add}(y|y_S) & \text{otherwise} \end{cases} \tag{4.8}$$

h_{cea} treats actions as *rules* of the form $o : x', z \to x$, where o is the operator (action) and x' is the precondition value of variable X which transitions to $X = x$ once the rule is applied. z is the set of preconditions on *other* variables which must also hold true in the current state prior to the rule being applied. These rules are applied automatically when the state meets the required preconditions.

Like h_{add}, h_{cea} operates recursively, as defined in Equation 4.10 and 4.9. Starting from state, S, in which variable $X = x'$ (denoted $S[x']$), all operators which achieve x are expanded in order to determine the cost of achievement, in much the same way as h_{add} and h_{cg} do. However, unlike these, the cost of achieving the associated preconditions is evaluated in the state resulting from the application of o. That is, instead of considering each precondition of o as independent of one another, the cost of achieving each $z_i \in z$ is computed only after x has been achieved. The state resulting from this application is referred to as a *context* state, as defined in Equation 4.11, where $s(x''|x')[z, x][y_1, ..., y_n]$ indicates the state s resulting from the achievement of x'' starting at value x', in which preconditions x and z are set. $y_1, ..., y_n$ are the effects of other applicable rules which are applied simultaneously once o is known to be the best cost minimising operator which achieves x.

In this form, this approach would result in an explosion of context states, making computation intractable. Therefore, once Equation 4.11 is applied, all information relating to variables other than X is discarded.

$$h_{cea}(S) = \sum_{x \in G^*} h_{cea}(x|x_S) \tag{4.9}$$

$$h_{cea}(x|x') = \begin{cases} 0 & \text{if } x = x' \\ \min\limits_{o:x'',z \rightarrow x} \left(1 + h_{cea}(x''|x') + \sum\limits_{x_i \in z} h_{cea}(x_i|x'_i) \right) & \text{if } x \neq x' \end{cases} \tag{4.10}$$

$$S(x|x') = \begin{cases} S[x'] & \text{if } x = x' \\ S(x''|x')[z, x][y_1, ..., y_n] & \text{if } x \neq x' \end{cases} \tag{4.11}$$

The additive nature of h_{cea} and the relaxations applied to contexts means that, like h_{add} and h_{cg}, the heuristic is inadmissible. In planning, this can lead to extreme over-estimations of the true goal distance, which could cause search to fail.

However, in the case of IGRAPH the majority of estimates required will be for the single literal goals contained in $\mathcal{G}$, making the heuristic potentially more accurate than h_{max} or h_{ff}. Table 4.2 again shows results of the heuristic estimate over a set of increasingly distant goals in the Rovers domain (which has largely disconnected goals), while Figure 4.7 visualises this. However, unlike Table 4.1, the estimates for all 40 problems are shown, in order that the difference with h_{ff} are clearer. This table further demonstrates that h_{cea} is inadmissible for most problems (particularly those with multiple goals), and also that h_{ff} often underestimates the true goal distance as solution length increases.

Using Helmert's Translator for Recognition

The translator developed by Helmert [78] is designed to transform a PDDL domain into an equivalent SAS$^+$ representation. However, as this is intended for use in planning applications [79], the output generated is optimised such as to allow for faster heuristic search by minimising the state-space. For example, if a problem has variables $\phi(V) = v_1 ... v_n$, but only v_1 appears in the goal G^*, and only v_2 is required to achieve this, variables $v_3 ... v_n$ will be pruned from the resulting output. This results in fewer actions, DTGs and a smaller causal graph, all of which speed up the planning process.

Clearly, these optimisations are of little assistance in a recognition context, where G^* is unknown and can be any combination of literals. Therefore, the translator has been modified to eliminate these optimisations. Concretely, this means that variables and the associated domains are never pruned from the output, and mutually-exclusive goals are allowed. In this latter case, as the translator still expects to parse a goal specification, this is randomly generated by IGRAPH, in order that every object present in the original PDDL file is present in at least one of the goal-literals passed to the translator. This forces the translator to generate variables for all objects in the domain, even though these may not all strictly be required for planning.

4.3.4 Goal Recognition with Suboptimal Heuristics

With any of the suboptimal heuristics described above and the other aspects of the relaxed model, everything required to perform goal recognition is now present. This amounts to using the observed evidence to construct a *maximum likelihood* Bayesian posterior probability across each sub-goal-space, before using Algorithm 3 to select those goals which are deemed to have had the most work put towards their achievement.

However, as the agent and observer in the relaxed model now cannot be guaranteed to have optimal heuristic estimates to a goal, and that only single literal goals exist explicitly in the goal-space, the value for $W_{ML}(G)$ may be

Problem	h_{max}	h_{ff}	h_{cea}	$\|P\|$	**Problem**	h_{max}	h_{ff}	h_{cea}	$\|P\|$
1	4	10	9	10	**21**	4	61	88	62
2	3	9	7	8	**22**	4	79	140	70
3	4	11	11	11	**23**	4	80	145	90
4	3	10	10	8	**24**	5	92	149	129
5	4	20	21	22	**25**	3	42	73	30
6	4	30	32	36	**26**	4	65	100	64
7	3	15	15	18	**27**	4	87	132	100
8	4	28	24	26	**28**	4	85	146	76
9	4	27	33	31	**29**	4	66	124	62
10	3	33	30	35	**30**	4	109	181	126
11	5	30	48	32	**31**	4	104	192	136
12	4	23	22	19	**32**	4	150	253	173
13	4	42	60	46	**33**	4	191	333	261
14	4	31	28	31	**34**	4	148	254	148
15	3	37	30	42	**35**	5	267	479	368
16	3	36	39	41	**36**	5	185	332	246
17	4	42	63	50	**37**	5	209	270	377
18	4	43	43	46	**38**	4	228	496	300
19	4	70	113	69	**39**	5	292	517	380
20	4	83	153	99	**40**	4	273	542	356

Table 4.2: Heuristic estimates from the initial state for the h_{max}, h_{ff} and h_{cea} heuristics on the IPC5 variant of the ROVERS domain.

uninformative. As the *work* performed is computed in the context of the entire observed plan, posteriors for one of the single literal goals which make up the true goal may be low, leading to an incorrect hypothesis or a late-commitment to the true goal.

The following section presents two methods of mediating this decreasing *work* function by first considering work performed in the context of causally-linked observations, and secondly by taking a *per observation* approach to how helpful an action is.

4.3.5 Work Performed in Relaxed Goal-Spaces

In Section 3.3, an optimal heuristic was used in the complete goal-space to provide a means of determining how much work, $W(G)$, had been expended on achieving a goal. Equation 3.1 defined this as the difference between the current and initial heuristic estimates to G, which was in turn relaxed to form Equation 3.2 (page

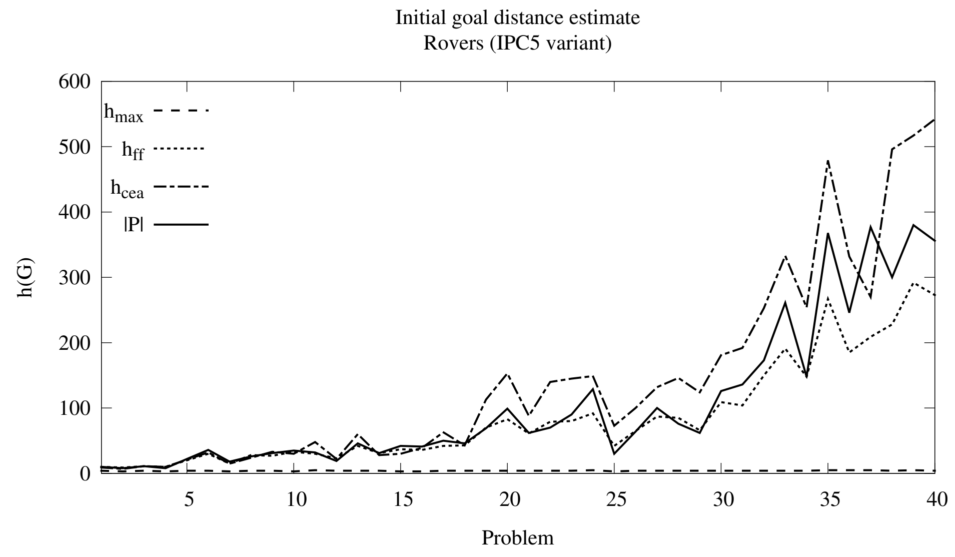


Figure 4.7: A plot of the various heuristic estimates shown in Table 4.2. Values for $h(G)$ are computed from the initial state in each respective problem of the Rovers domain from IPC5. The plan length, $|P|$ is taken to be the shortest solution generated by a planner in this same competition. The inadmissibility of the h_{cea} heuristic is clearly visible, as most problems have a higher initial estimate than final plan length. Similarly, h_{ff} overestimates some problems, but in general underestimates the distance to the goal. Only h_{max} is admissible, but provides extremely poor estimates.

52). This considered suboptimal plans being observed, by stating that the work put towards a goal is equal to the number of observations which have lowered the initial estimate, divided by the total number of observations. That is, the *maximum likelihood* of each observation contributing towards the goal, $W_{ML}(G)$.

The principle behind using the maximum likelihood to compute the work performed by an action still holds in the new relaxed goal-space, but crucially, only when the agent's goal is a single literal $G^* \in \pi(F)$. If this assumption does not hold — which it rarely does — any estimates computed for conjunctive goal G^* will be *biased* towards the order in which each individual goal is achieved. For example, if $G^* = \{A, B\}$, but that every action possible is guaranteed to only contribute towards achieving A *or* B, then the order of actions will affect the posterior probabilities of each goal if the maximum likelihood model is used.

Table 4.3 demonstrates how the value of $W_{ML}(A)$ decays over time once A has been achieved by observation O_3^A. Goal B is also affected by having an artificially low value of $W_{ML}(B) = 0.25$ after action O_4^B has been observed.

Obs	P(A)	P(B)	$W_{ML}(A)$	$W_{ML}(B)$	P($A\|O$)	P($B\|O$)
O_1^A	0.50	0.50	1.00	0.00	0.83	0.17
O_2^A	0.83	0.17	1.00	0.00	0.88	0.12
O_3^A	0.88	0.12	1.00	0.00	0.89	0.11
O_4^B	0.89	0.11	0.75	0.25	0.84	0.16
O_5^B	0.84	0.16	0.60	0.40	0.77	0.23
O_6^B	0.77	0.23	0.50	0.50	0.68	0.32

Table 4.3: The prior and posterior values for goals A and B when the true goal is $G^* = \{A, B\}$, and both literals are treated individually, as opposed to in the correct conjunctive form. Equation 4.21 is used to compute the posteriors, in order that non-zero values are produced. Of the six observations, the first three only assist in achieving A, while the last three only assist in achieving B. Note that the posterior for A decays after observation O_4^A, due to a similarly decaying value for $W_{ML}(A)$. In contrast, the posterior value for B at the end of the plan is much lower than that of A due to the maximum likelihood work function not recognising that the first three steps were of no relevance in achieving B.

Clearly having a work function which demonstrates this behaviour for conjunctive goals that can be achieved at different timesteps is of little use in practical recognition. Thus, the relaxed goal recognition model must offer a viable alternative to the maximum likelihood version of determining work performed.

Achieving Goals Throughout Observation

In the original goal-space representation, $\mathbb{G}$ was guaranteed to contain an entry for all states $S \in \mathbb{S}$ in addition to all partial states/goals. The relaxed goal-space now *only* models probability distributions across each mutex set, which is identical to modelling the probability of a *state* being the goal, rather than a subset of the final-state. The implications of what this change means to goal recognition is considered in Chapter 5.

While the relaxed goal-space closely models state-estimation, it is still skewed towards the assumption that every step in the plan will contribute to the goal being achieved in the final plan step. That is, if a goal is *not* converged upon after an observation it can be eliminated. In essence, every plan step will contribute towards the complete achievement of the goal after the final observation, at which point the plan will cease. This assumption which has featured prominently in the literature [18, 38, 97, 111], is severely limiting and forms one of the motivating factors for a new model of goal recognition.

In planning, the intuition behind heuristic-guided search is that the heuristic will always try to minimise the cost of achieving the goal. However, if goals are achieved *throughout* plan formation, then the heuristic must also take care not to negate early achievements in forming a plan for the remaining goals. Equation 4.12 states that any literals which have remained true over the previous two observations ($h(G) = 0$) have had work put towards *maintaining* their achievement. The rationale for this is that if an agent keeps a fact true over $n \geq 1$ consecutive timesteps, then this fact may be a part of the final goal and should be treated accordingly. This is referred to as giving "stagnant" facts a *heuristic bonus*. In practice, this bonus amounts to considering actions which have not interacted with a goal's heuristic estimate (which has been zero for the previous two timesteps), to have been *helpful* in achieving the goal, in much the same manner as an action is helpful to an unachieved goal if it lowers the heuristic estimate. Equation 4.12 shows how each observation is classed as helpful or not.

$$H_t(G) = \begin{cases} 1 & \text{if } h_t(G) < h_{t-1}(G), \\ 1 & \text{if } h_t(G) = h_{t-1}(G) = 0, \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

The application of a heuristic bonus to goals which are achieved prior to plan completion is a necessity in the relaxed goal-space. Without the bonus, and considering only the heuristic distance moved towards achievement since observation began, the probability of the goal will decrease monotonically until plan termination. This is simply because the assumption of the agent always working towards a single (conjunctive) goal no longer holds.

However, while giving bonus scores to facts which are true across consecutive states allows those facts to remain probable goal candidates, it *obscures* those goals that are both mutually-exclusive and are still being actively pursued. That is, if goal A has been true for several observations, but that goal B is moving closer to achievement, the addition of a bonus for goal A will result in both goals receiving higher probabilities. Over time this will cause goal A to always be output as the true goal, despite goal B consistently becoming closer. B will only be considered as the true goal once it has become true and A has been negated, and crucially, *the number of helpful or bonus observations for B exceeds those of A*.

Unfortunately, there is no elegant solution to this problem. In the example given, it is stated that B is the true goal, but the heuristic movement could also be observed through the side-effects of achieving the non-mutex goal C.

Therefore, the most reasonable course of action is to use the final observation as an indicator that A truly was not the goal. This uses hindsight to note that the amount of work put towards A was lower than previously considered. Equation 4.17 shows how this affects the *maximum likelihood* of a literal being part of the true goal. In this, a bonus is only applied to the member of a sub-goal-space which is *currently* true and has been consecutively true for more than one observation.

Equation 4.15 defines when the bonus is applicable to a goal, G, which is a member of sub-goal-space $\mathcal{G}$, and where $C(G)$ refers to the number of consecutive timesteps that a fact has been true (Equations 4.13 and 4.14). This is used by Equation 4.16, which defines whether an observation at time t is helpful to the goal's achievement through either a reduction in the heuristic estimate or the bonus, $B(G)$, being applicable. This progresses the concept of *work* performed by the agent from being one of counting heuristic reductions alone, to one of counting "helpful" observations (reductions and bonuses), as defined in Equation 4.16.

$$C(G) = C(G, S_{current}) \tag{4.13}$$

$$C(G, S_t) = \begin{cases} 1 + C(G, S_{t-1}) & \text{if } G \in S_t, \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

$$B(G, \mathcal{G}) = \begin{cases} 1 & \text{if } C(G) > 1, \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

$$H_t(G, \mathcal{G}) = \begin{cases} 1 & \text{if } h_t(G) < h_{t-1}(G), \\ 1 & \text{if } B(G, \mathcal{G}) > 0, \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

$$W_{ML}(G) = \frac{\sum_{t=1}^{|O|} H_t(G, \mathcal{G})}{|O|} \qquad \text{where } G \in \mathcal{G} \tag{4.17}$$

However, the assumption that the currently true literal in a sub-goal-space is the true goal is unreasonable for most domains, particularly those with high concurrency in plans. The constraints under which the bonus is applied are therefore tightened to only consider facts which are both true for at least two timesteps and for which all other members of the sub-goal-space have had their heuristic estimate increase. Thus, in the previous example the bonus would never be applied as B is constantly pursued after A has been achieved. Equation 4.18 shows this updated version of $B(G, \mathcal{G})$.

$$B(G, \mathcal{G}) = \begin{cases} 1 & \text{if } C(G) > 1 \text{ and } \forall G_i \in \{\mathcal{G} \setminus G\}, h_t(G_i) > h_{t-1}(G_i), \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

To recap, the above equations define a method of counteracting the loss of completeness in the relaxed goal-space, caused by suboptimal observations and concurrency in achieving goals. A *bonus* is applied to currently true goals where an observation has not reduced the distance to any mutex goal in the respective sub-goal-space. This is done to mitigate situations in which the true goal has been achieved, but other members of the same sub-goal-space may become closer through side-effects of future observations, or suboptimal choices by the subject.

While observations which do not reduce the distance to a goal literal can now be classed as helpful to their achievement, the use of the bonus score is not without issue. By believing that the currently true literal G in a sub-goal-space is the true goal, there is potential for hypotheses to become skewed towards currently true goals only. This perhaps belies a more fundamental question of whether a literal which has been true for n observations is a more relevant goal candidate than the subsequent transition to a mutex literal, G' at time $n + k$. The use of the bonus count in producing an estimate for $W_{ML}(G)$ states that until further evidence is observed, there is no reason to believe that any goal other than G is the goal (unless there has actually been more work put towards a mutex goal). Only after G' has been achieved can the work function be updated to remove these incorrect bonus scores, thus reducing the work associated with G being the true goal.

A positive side-effect of applying a bonus to stagnant facts is that the model can implicitly detect *persistent* goals. These are goals which are true in the initial state and remain unmodified throughout observation. The detection of these is highlighted as one of the features the heuristic-based model has in common with the prior work of Geib *et al.* in Section 3.6.2, where they were referred to as *safety* goals.

4.3.6 Alternatives to Maximum Likelihood

By altering the conditions under which an observation is classed as *helpful* to a goal, the modified maximum likelihood function advances towards a more accurate approximation of the original definition of W_{ML} (page 52). However, it may be that the conditions under which the *bonus* is applied are never met, as having *all* mutex goals in a sub-goal-space become further away after an observation may not occur in some domains. It is more likely that one or two members will have some slight movement over time, but that the currently true goal remains achieved throughout the remainder of observation.

The key here is that once a goal is achieved (and is assumed to remain achieved), all subsequent observations are irrelevant to the amount of work performed if the bonus is not applied. This is especially true if there is no *causal link* between the achievement of the first goal and subsequent goals. The goals are therefore at least semi-independent of one another, and may have weak constraints on their temporal ordering. It is therefore logical that recognition performance could be improved if these irrelevant observations were ignored from the computation of work performed. The following section explores one approach to determining which observations contribute towards the achievement of a goal, and uses this in computing a more useful value for $W(G)$.

Work Performed Across Sub-Plans

In the complete goal-space, the value of W_{ML} is a perfect indicator of whether an individual or conjunctive literal is a goal candidate. In the relaxed model this is not the case. Here, the agent's goal is considered to be made of several individual goal-literals, which is not equivalent to a conjunctive goal in the original goal-space. Therefore, the agent's plan may achieve one goal with the first part of the plan, and the remaining goals with the remainder of the plan. Alternatively, the agent may achieve goals using any ordering of actions, with some actions contributing towards achieving more than one goal literal. The challenge here is in modelling the *causal-links* between observations, such that only those actions which were required to precede the helpful observation are considered relevant to computing the *work* performed. This can be viewed as being equivalent to the construction of a graph which models these causal-links between observed actions and the states which precede and succeed them.

Consider the following logistic example shown in Figure 4.8, in which three packages, `package1, package2, package3`, must be delivered to their respective destinations. All packages start out at the same location, `loc0`, with two trucks `truck1, truck2`, also at this location. `package1` and `package2` must be delivered to location `loc3` and `package3` to `loc6`. The recogniser then observes the twelve-step, totally-ordered plan shown in Table 4.4 which achieves the agent's true conjunctive goal of `(and (at package1 loc3) (at package2 loc3) (at package3 loc6))`.

A human observer can see that there are two sub-plans within this totally-ordered plan and use this information to infer each *sub-goal*, along with the true overall goal. Here, each sub-plan is referred to as a *plan thread* as it will normally account for only part of the overall plan, and may interleave with other plan threads when necessary. The resulting graph[10] is referred to as the *plan-thread graph*. Extracting the relevant portions of the plan-thread graph should allow a more intuitive value for $W(G)$ to be extracted. The graph is constructed iteratively as follows.

Plan Thread Graph Construction After each observation O the observed plan is scheduled such that each observation is assigned a timestamp s indicating its earliest possible application time. That is, the observation *could* have been seen at time $0 \leq s \leq t$, but due to the linear nature of the observed plan, it was observed at time t. The simple algorithm which performs this scheduling is omitted here for brevity, but can be found in Appendix A.

The *scheduled observation* O_t^s is then linked to at least one of the states in the plan graph which exist at time s. Scheduled actions are simply a tuple $O_t^s = \{s, O_t\}$, corresponding to an action observed at time t with earliest scheduled time s. While plan threads are constructed iteratively, they can also be extracted by performing an all-paths search from any leaf nodes to the root of the graph (which is equivalent to the initial state of the problem). The set of existing plan threads is denoted $\mathcal{T}$, and is a subset of the original plan-space, $\mathcal{T} \subseteq \mathbb{P}$, which is updated after each observation.

Once the plan has been scheduled, the construction of the plan-thread graph can begin. The output of this process is a series of *plan threads*, each of which is itself a normal, valid plan as defined in Section 3.2.3. Each plan thread encapsulates the set of actions which are required in order for the *last* action in the plan thread to be applicable. Any actions which are part of the original plan, but have no effect on the achievement of this final scheduled action will not appear in the

[10]In reality this may actually an *n-ary tree*, provided that each action requires a single predecessor state. If more than one is required, it can be considered a single-directed graph.

t	s	*Obs*
1	0	`load package1 truck1 loc0`
2	0	`load package2 truck1 loc0`
3	1	`drive truck1 loc0 loc1`
4	0	`load package3 truck2 loc0`
5	2	`drive truck1 loc1 loc2`
6	1	`drive truck2 loc0 loc4`
7	3	`drive truck1 loc2 loc3`
8	4	`unload package1 truck1 loc3`
9	2	`drive truck2 loc4 loc5`
10	5	`unload package2 truck1 loc3`
11	3	`drive truck2 loc5 loc6`
12	4	`unload package3 truck2 loc6`

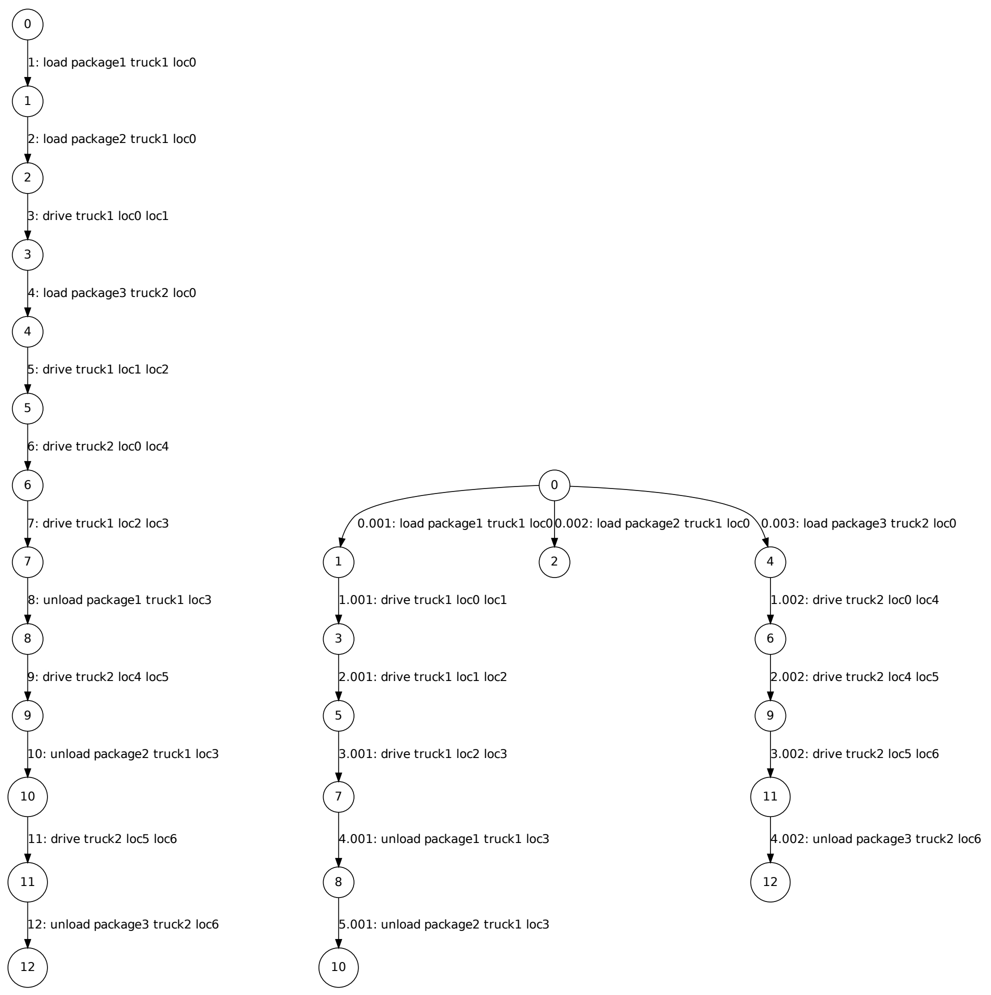
Table 4.4: The timestamps for a series of observed actions in both a totally-ordered and scheduled context. Column t indicates the time at which the action was observed, while column s indicates the earliest time at which the action could have been observed. The scheduled time can then be used in construction of the plan-thread graph. An algorithm which performs this scheduling is given in Appendix A.

plan thread.

Definition 22. Plan Thread
A *plan thread* is a tuple $T = \{I, \Theta, \mathit{head}(T)\}$ where I is the initial state of the problem and Θ is a set of ordered tuples representing the actions and state observed thus far. Each $\theta \in \Theta$ is itself a tuple $\theta_t = \{S_{pred}, O_t^s, S_{succ}\}$, where S_{pred} is the set of predecessor states which are required to apply action O_t^s, such that $\bigcup S_{pred} \supseteq O_{pre}$. S_{succ} represents a single successor state which is formed by applying actions $O_1^1 ... O_t^s$ in order from I. Each $\theta \in \Theta$ is naturally ordered by the timestamp $s \in \mathbb{Z}^+$ associated with each observation, $\theta(O_t^s)$.

head(T) represents the *plan thread head* — the state which is created by applying all actions in the thread in order from state I, and is equivalent to the final member of Θ, $\mathit{head}(T) = \Theta_{last}(S_{succ})$.

Plan threads are constructed iteratively after each observation, with a single plan thread guaranteed to be returned during the threading process. Whether this is an existing thread or a new thread is dependent upon the timestamp assigned to the observation by the scheduler and the thread to which it is assigned. If the observation has timestamp $s \geq \mathit{head}(T)_t$, where $\mathit{head}(T)_t$ is the timestamp


0
1: load package1 truck1 loc0
1
2: load package2 truck1 loc0
2
3: drive truck1 loc0 loc1
3
4: load package3 truck2 loc0
4
5: drive truck1 loc1 loc2
5
6: drive truck2 loc0 loc4
6
7: drive truck1 loc2 loc3
7
8: unload package1 truck1 loc3
8
9: drive truck2 loc4 loc5
9
10: unload package2 truck1 loc3
10
11: drive truck2 loc5 loc6
11
12: unload package3 truck2 loc6
12
0
0.001: load package1 truck1 loc0
0.002: load package2 truck1 loc0
0.003: load package3 truck2 loc0
1
2
4
1.001: drive truck1 loc0 loc1
1.002: drive truck2 loc0 loc4
3
6
2.001: drive truck1 loc1 loc2
2.002: drive truck2 loc4 loc5
5
9
3.001: drive truck1 loc2 loc3
3.002: drive truck2 loc5 loc6
7
11
4.001: unload package1 truck1 loc3
4.002: unload package3 truck2 loc6
8
12
5.001: unload package2 truck1 loc3
10

(a) The original, totally-ordered plan.

(b) The same plan as Figure 4.8a after being transformed into a plan-thread graph.

Figure 4.8: The plan shown in Table 4.4 in its original totally-ordered form (4.8a) and after scheduling and thread-graph construction has taken place (4.8b).

of the head of T, then it will be appended. If $s < head(T)_t$, the thread will be branched at time s to form a new thread, T_{new}.

- Append — An action O^s which has been scheduled at timestamp s can be appended to $T \in \mathfrak{T}$ if s is equal to the timestamp of $head(T)$ and $O^s_{pre} \subseteq head(T)$. This will result in a tuple $\theta_{new} = \{head(T), O^s, head^{s+1(T)}\}$, where $head^{s+1}(T)$ is the state formed by applying O^s in the head state. After θ_{new} is created, it is added to Θ, and $head(T)^{s+1}$ becomes the new thread-head.

The number of threads present in the graph remains the same, $|\mathcal{T}_t| = |\mathcal{T}_{t-1}|$.

- Branch — Given a scheduled observation O^s_t, an existing thread $T \in \mathcal{T}$ can be *branched* to form a new thread T_{new}, if $\exists\theta \in T(\Theta)$, such that $\theta(S^s_{succ}) \subseteq O_{pre}$. That is, thread T contains a tuple θ which has a successor state S_{succ} that is true at time s, which satisfies all preconditions of O^s_t. The branched thread is created by copying the actions of thread T up to time s. Formally, $T_{new} = \langle T(I), T(\Theta^s), head(T)^{s+1}\rangle$, where $T(\Theta^s) = \langle\theta_0 ... \theta_s\rangle$. This new thread then has O^s *appended* in the manner described above. Thread T remains unaltered by this process, with $\mathcal{T}_t = \mathcal{T}_{t-1} \cup T_{new}$

Both of the above operations assume that there will be a single thread which the observation can be appended to at time s. However, for most plans this is unlikely, as there is often a degree of overlap between threads, meaning they must be *merged* before the action becomes applicable. For example, thread T_A may have a head containing facts $\{x, y\}$, while thread T_B has head $\{y, z\}$. A new observation which requires $\{x, y, z\}$ to be true will require a combination of both thread-heads, meaning threads T_A and T_B must be merged to form a new thread before the observation can be appended, with the same principle applying to branching of threads at time s.

Merging Threads If an action cannot be appended or branched from any of the existing thread heads, it is because the action's preconditions cannot be met by a single state at time s. In this case, multiple threads must be merged together in order that the action's preconditions can be met. Given that each thread represents a partial-plan, merging $2 \leq n \leq |\mathcal{T}|$ threads is guaranteed to produce the true current state.

The specific n threads selected to be merged are those which last achieved the required preconditions. Preferring late-achievers of a fact over early-achievers has two advantages. First, this information has already been determined separately during the initial scheduling process, and can be directly referenced during threading. Secondly, this removes the potential for *ties* between threads as to which is the best for inserting the observation into. By always preferring the *last* action which achieved a fact f, the algorithm is certain to terminate. If a thread, T_{early}, which contains an earlier achiever of f is used instead, there is a possibility that a future observation which is appended to T_{early} will delete f within the associated thread-head. This will render threading impossible because the union of

Algorithm 7 The algorithm for updating the current thread-head is simply a matter of taking the thread T, and applying the operator O_t^s which is associated with each tuple θ in the totally-ordered set Θ.

```
Require: $T = \{I, \Theta, head(T)\}$
  $head(T) := I$
  for all $\theta \in \Theta$ do
    $a := \theta(O^s)$
    $head(T) := (head(T) \setminus a_{del}) \cup a_{add}$
  end for
```

the active thread-heads no longer contains the relevant set of preconditions, and it cannot be detected by the scheduler, as the action has not yet been observed.

When n threads must be merged, a new thread is constructed by simply copying over all tuples within each thread's Θ set into a new set Θ_{merge} which represents the merged thread. This new tuple T_{merge} now represents a single thread, but still retains the parallelism which was present in the previous threads, as each $\theta \in \Theta_{merge}$ is still valid. That is, each tuple encapsulates the states required for an action to be applicable, plus the single successor state. In this way, merged threads can themselves contain multiple previously-merged threads.

Once Θ_{merge} is constructed from the set of threads to merge, $\mathcal{T}_{merge}$, as $\Theta_{merge} = \bigcup_{i=1}^{|\mathcal{T}_{merge}|} T_i(\Theta)$, a further *stub* tuple θ_{merge} must be appended to construct the state from which O_t^s is applicable. The purpose of this tuple is to provide a link between the n threads to be merged, such that $\theta_{merge} = \{S_{merge}, M, head(T)^s\}$, where $S_{merge} = \bigcup_{i=1}^{|\mathcal{T}_{merge}|} head(T_i)$, M is an empty action of the form $M = \langle \emptyset, \emptyset, \emptyset \rangle$, and $head(T)^s$ is computed as in Algorithm 7. Finally, with the merge action in place and the new plan-head computed, action O^s can be appended as normal.

Threading Algorithm After each observation at time t, the action is added to the totally-ordered plan P, which is then scheduled using Algorithm 13 (page 215). As the scheduler is deterministic and considers actions strictly in the order in which they were observed, they will always be assigned the same scheduled timestamp s. The timestamped action, O^s, and the set of threads, $\mathcal{T}$, which existed at time $t-1$ are then passed to Algorithm 8 in order that the observation can be assigned to a thread.

The algorithm returns a single thread T_{new} which has its head at time $s+1$, mapped to the threads from which it was created. These modified threads can be discarded when performing *batch-threading* (wherein the entire plan is known

Algorithm 8 The *batch* threading algorithm. This requires a pre-scheduled plan $P_S = \langle O_0^0 ... O_n^m \rangle$, where n is the total number of actions in the plan and m is the timestamp of the latest scheduled action.

```
Require: $P_S$
  $t := 0$ {Initialise timestamp counter}
  $T_{initial} := \langle I, \emptyset, I \rangle$
  $add(\mathfrak{T}, T_{initial})$
  while $|P_S| > 0$ do
    $\mathfrak{T}_{old} := \emptyset$
    $\mathfrak{T}_{new} := \emptyset$
    for all $O^s \in P_S$ do
      if $O^s(s) = t$ then
        $\mathfrak{T}_{parent} := \emptyset$
        $T_{new} := scheduleAction(O^s, \mathfrak{T}, \mathfrak{T}_{parent})$
        $add(\mathfrak{T}_{old}, \mathfrak{T}_{parent})$
        $add(\mathfrak{T}_{new}, T_{new})$
      end if
    end for
    $t = t + 1$ {STRIPS-based actions, all have cost 1}
    $\mathfrak{T} := (\mathfrak{T} \setminus \mathfrak{T}_{old}) \cup \mathfrak{T}_{new}$
  end while
  return $\mathfrak{T}$ {Return final threads at $t$}
```

apriori), or retained in *iterative-threading*. In the former case which is detailed in Algorithm 8, all actions are processed at-once, meaning that only a single set of threads need be kept during processing as all actions which will exist at time $0 \leq s \leq |P|$ are known. This essentially removes the need for explicit *branching* of threads.

In the latter case of iterative threading (Algorithms 9 and 10), all threads present at time s must be retained throughout execution. By keeping note of all threads which exist at each timestamp, the observed action can be appended to one or more of these threads without the need to consider actions before or after time s. For instance, if a new observation is scheduled for time $s = 5$, all threads which have their thread-head at time $0 \leq k \leq s$ need be considered as potentially supporting the observation (which leads to appending, branching or merging).

Once threading is complete, the thread to which the observation has been appended is returned to IGRAPH. This provides context as to which of the previous observations were required for the current observation to take place. This information can then be used to assist in the recognition process, by deriving a new *work* function.

Algorithm 9 The *iterative* threading algorithm. This requires a single scheduled observation O^s which is to be added to at least one of the threads active at time $0 \leq s \leq m$, and a list of all threads which have existed at all timepoints during the threading process, $\mathfrak{T}_{all} = \{\mathfrak{T}_0 ... \mathfrak{T}_m\}$, where m is the timestamp of the latest thread-head in $\mathfrak{T}_{all}$.

```
Require: $O^s, \mathfrak{T}_{all}$
  $\mathfrak{T}^s := \mathfrak{T}^s_{all}$
  $T_{new} := scheduleAction(O^s, \mathfrak{T}^s)$ {See Algorithm 10}
  $add(T_{all}(s+1), T_{new})$ {Add the new thread to those at time $s+1$}
  return $T_{new}$
```

Algorithm 10 The *scheduleAction* method takes in a single action to be appended to one of the threads in $\mathfrak{T}^s$ or a combination of threads within this. The collection $\mathfrak{T}_{old} = \emptyset$ is populated with those threads which are used to create the returned value, T_{new}, such that $\mathfrak{T}_{old} \subseteq \mathfrak{T}^s$.

```
Require: $O^s, \mathfrak{T}^s, \mathfrak{T}_{old}$
  $\mathfrak{T}_{link} := \emptyset$
  for all $pc \in O_{pre}$ do
    $T_{link} := lastAchiever(pc, \mathfrak{T}^s)$
    $add(\mathfrak{T}_{link}, T_{link})$
  end for
  $T_{new} := NULL$
  if $|\mathfrak{T}_{link}| > 1$ then
    $T_{new} := merge(\mathfrak{T}_{link})$ {Merge and append “stub” action}
  else
    $T_{new} := poll(\mathfrak{T}_{link})$ {Only 1 thread, so remove from list}
  end if
  $append(O^s, T_{new}$ {Add the action to the chosen thread}
  $\mathfrak{T}_{old} := \mathfrak{T}_{link}$
  return $T_{new}$
```

Using Plan Threads to Compute Work Performed Now that a more accurate representation of the causal links between actions has been produced, it becomes possible to obtain a more refined value for $W(G)$. Given that the observed action O_t lowered the heuristic estimate to G, the thread to which this was appended is marked as being *helpful* in achieving G. That is, instead of computing $W_{ML}(G)$, which considers those steps that have been useful across the entire observed plan, the *threaded maximum-likelihood* score $W_{MLT}(G)$, considers those observations which were helpful in achieving G against *only* those observations which were required to make these actions applicable. For example in Figure 4.9, if observation O_6 is considered helpful in achieving goal G, then only observations

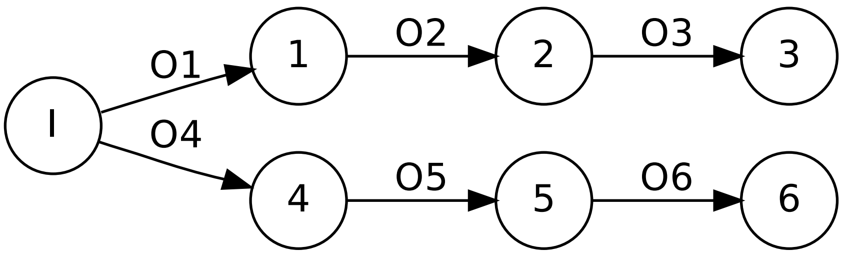


Figure 4.9: The result of applying Algorithm 8 to the observed, totally-ordered plan $P = \langle O_1 ... O_6 \rangle$ shown in Table 4.3 (page 96). Actions which can be scheduled at a timestamp earlier than their observation time are linked only to the actions/states which achieve their preconditions. For instance, O_4 is not linked to O_3, as this latter observation has no impact upon the achievement of O_4. This results in two sub-plans being formed, which can be used to improve the value for $W(G)$.

4 and 5 are required to compute the value of work performed, as these were the only actions which allowed the execution of observation 6.

In simple terms, the threaded maximum-likelihood value is the same process as computing W_{ML}, but in the context of a (hopefully) smaller set of observations, such that $W_{MLT}(G) \geq W_{ML}(G), \forall G \in \mathcal{G}$. Equation 4.9 provides a formal definition of this process, where $\mathcal{T}_{helpful} \subseteq \mathcal{T}$ is the set of threads which contain an action that has been considered helpful in achieving fact G. Note that there is still potential for the bonus to be applied through $H(G, \mathcal{G})$, which remains as defined in Equation 4.16. The only difference between this and W_{ML} is the denominator, which considers a smaller or equal number of observations as being relevant.

$$W_{MLT}(G|O_t) = \frac{\sum_{i=1}^{t} H_i(G, \mathcal{G})}{\sum |T|} \forall T \in \mathcal{T}_{helpful}(G), \text{where } G \in \mathcal{G} \tag{4.19}$$

Here the numerator remains the same as in the original formula for W_{ML} — representing the number of observations which have lowered the estimate to G. The denominator represents the cumulative length of every thread which has been helpful towards achieving G. Steps which are not relevant to the achievement of G are ignored from the calculation, resulting in a higher score.

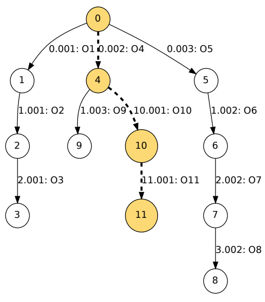
In the previous example using Figure 4.9, the value of work performed after observation $O^{s=3}_{t=6}$ would therefore be $W_{MLT}(G^*) = \frac{1}{3}$, given that only this observation was helpful; there are no bonuses applied, and the thread the observation is appended to has length 3. If instead observations $O^{s=1}_{t=2}$, $O^{s=2}_{t=3}$ and $O^{s=3}_{t=6}$ were helpful, the value would be $W_{MLT}(G^*) = \frac{3}{6} = 0.5$, as both threads contain a helpful action. In this case, $W_{MLT}(G^*) = W_{ML}(G^*)$. A visual example of this is shown in Figure 4.10, in which the thread-graph becomes more connected as further observations are processed, until it represents a single-thread.

As this figure shows, depending upon the domain and plan observed, the thread graph can become highly-connected as more observations are processed. Therefore, the longer the plan the more likely it is that observations which are present in multiple threads become necessary to activate subsequent actions which lower the distance to the goal.

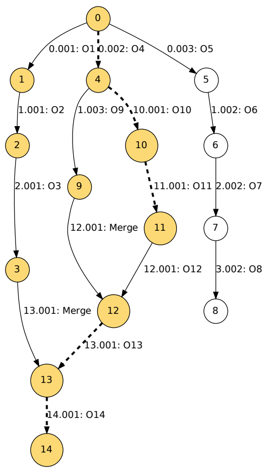
***All False* Goals** As with the maximum-likelihood work function, the threaded-maximum-likelihood must also accommodate the *all false* literals introduced in Section 4.3.1. Luckily, this is trivial as the work contributed towards an *all false* goal within a sub-goal-space remains the same as for the case of maximum-likelihood. The threaded-maximum-likelihood for a normal, positive goal is derived as a function of how many steps have been helpful over those steps *required* for them to be helpful. However, the work put towards the *all false* goal remains the number of observations which have moved *all* positive, mutex goals further from being achieved. The thread to which these actions are appended is irrelevant.

Like the maximum likelihood model, W_{MLT} retains the assumption that all goals in the sub-goal-space are mutually-exclusive, despite this no longer being the case. It also retains the implicit assumption that the majority of steps in the plan or thread will contribute towards achieving the true goal G^* (with the hope being that $W_{MLT}(G^*)$ provides a higher value than simply $W_{ML}(G^*)$).

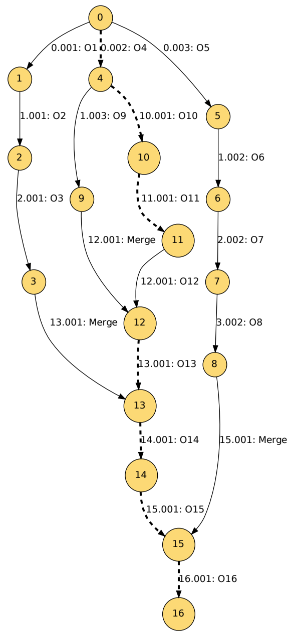
However, in domains where several overlapping goals are achieved using very short plan-threads, but the overall plan length is considerably higher, both of these work functions will produce values which degrade over time. The following section therefore presents an alternative model in which only the *current* observation is used in the computation of $W(G)$.



(a) A threaded representation of an incomplete plan.



(b) The threaded representation of the same plan, with additional observations.



(c) The threaded representation of the complete plan.

Figure 4.10: The thread-graphs produced at $t = 11$, $t = 14$ and $t = 16$ for an example 16-step plan. In all cases actions which have been helpful in achieving the goal G are denoted with a dashed line. Threads which contain at least one of these helpful observations are shaded. In Figure 4.10a, only a single thread is considered helpful and all observations within it are also helpful, giving a value of $W_{MLT}(G) = \frac{3}{3} = 1$. In Figure 4.10b, an action in the left-most thread has also been useful in achieving G, therefore it too is included as part of the sub-plan which is trying to achieve G. The value of work in this case is $W_{MLT}(G) = \frac{5}{10} = 0.5$. The respective values for Figures 4.10a and 4.10b using only *maximum likelihood* would be $W_{ML}(G) = \frac{3}{11} = 0.27$ and $W_{ML}(G) = \frac{5}{14} = 0.36$. In the final figure, all threads have converged into a single thread which encapsulates the entire observed plan, giving a value of $W_{MLT}(G) = W_{ML}(G) = \frac{7}{16} = 0.44$.

Work Performed Across Mutex-Sets

As stated above, when computing $W_{ML}(G)$ the entire observed plan is considered relevant, which may lead to low values during later observations as only single literal goals are considered. Threading plans can assist in mitigating the possibility of goals being pursued in an interleaved and partial-ordering by producing higher values than would otherwise be computed by W_{ML}. However, both of these cases ignore the fact that goals $G \in G^*$ are probably mutually-exclusive with several others in $\mathcal{G}$. For example, if an observation is helpful towards goal G_1, then it can be considered *unhelpful* for all those goals which are mutually-exclusive with G_1 (and have not also become closer).

Calculating work in this context takes full advantage of the relaxed multi-variate goal-space model $\mathcal{G}^V$, by considering how helpful each observation is with respect to only the members of each sub-goal-space. As each sub-goal-space $\mathcal{G}_i$ contains only facts which are mutually-exclusive, the amount of work performed by single action, W_{SA}, can be said to be *distributed* across these goals. Equation 4.20 provides a definition of this, where $\mathcal{G}$ is the goal-space of which G is a member and $\mathcal{G}^{nearer} \subseteq \mathcal{G}_i$ is the set of all goals in the goal-space which have had their heuristic estimate reduced by the observation or have had a bonus applied.

$$W_{SA}(G|O_t, \mathcal{G}) = \begin{cases} \dfrac{1}{|\mathcal{G}^{nearer}|} & \text{if } H_t(G, \mathcal{G}) > 0, \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

W_{SA} can be seen as a minimalist interpretation of the threaded-maximum-likelihood function presented previously. Here, instead of considering only those steps in the plan which are deemed relevant to the specific goal, only the current observation is of interest. W_{SA} spreads the work performed across those goals which have become closer through the previous observation, $W_{SA}(G) \in [0 : 1]$.

Table 4.5 shows the value of $W(G)$ using each of the proposed metrics after each observation, in the context of the example plans given in Figure 4.8.

Unlike W_{ML} and W_{MLT}, the *single action* configuration does not consider historical context. Moreover, it does not correct any historical references to observations which have been incorrectly assigned the heuristic *bonus*, as defined in Section 4.3.5. That is, if observation O_{t-1} is marked as helpful to a goal (where $t < |P|$) but this is later removed due to the goal estimate changing after a subsequent observation, the probability of the goal will *not* be recomputed to consider this — only the current value of W_{SA} (and thus whether the current observation

Time	`(at package1 loc3)`			`(at package2 loc3)`			`(at package3 loc6)`		
	W_{ML}	W_{MLT}	W_{SA}	W_{ML}	W_{MLT}	W_{SA}	W_{ML}	W_{MLT}	W_{SA}
1	1.00	1.00	0.14	0.00	0.00	0	0.00	0.00	0
2	0.50	1.00	0	0.50	1.00	0.13	0.00	0.00	0
3	0.67	1.00	0.33	0.67	0.67	0.33	0.00	0.00	0
4	0.50	1.00	0	0.50	0.67	0	0.25	1.00	0.13
5	0.60	1.00	0.50	0.60	0.75	0.50	0.20	1.00	0
6	0.50	1.00	0	0.50	0.75	0	0.33	1.00	0.33
7	0.57	1.00	1.00	0.57	0.80	1.00	0.29	1.00	0
8	0.63	1.00	1.00	0.50	0.67	0	0.25	1.00	0
9	0.56	1.00	0	0.44	0.67	0	0.33	1.00	0.50
10	0.50	1.00	0	0.50	0.71	1.00	0.30	1.00	0
11	0.45	1.00	0	0.45	0.71	0	0.36	1.00	1.00
12	0.42	1.00	0	0.42	0.71	0	0.42	1.00	1.00

Table 4.5: The difference in value for $W(G)$ using the various definitions, over the 12-step example plan outlined in Figure 4.8. Only the true goals are enumerated. Entries for W_{SA} with a '0' indicate that no work was performed for the relevant goal.

is *helpful*) is considered.

In some ways this makes the W_{SA} function a more naïve approach to computing the helpfulness of an observation, but in doing so better captures the interaction of the agent with multiple goals. Further, any historical mistakes by the agent are ignored, minimising the chance of probabilities decaying over time

While assigning bonuses to achieved facts is a suitable means of continuing to support potential goals after achievement, the relaxed model must be modified in one more way before recognition under relaxed conditions can take place. This involves accepting that both agents and observers are not rational, and therefore that they may make mistakes in their heuristic estimates. In particular, the potentially suboptimal nature of plans combined with the relaxed single literal goal-space can mean that an observation contributes no work towards a fact which is actually a goal. This, in turn, would produce a posterior probability of zero for the goal, which eliminates it from future consideration.

Preventing Zero-Valued Posteriors

The above modifications to the definition of work performed are in accordance with the assumption that the agent will display *bounded rationality* as outlined in Section 4.3.1. However, all of these functions (which are used as the *likelihood* function during Bayesian updates) have the potential to output a zero-value for work performed by an observation. In the complete and optimal goal-space this is not important — indeed, it is the preferred behaviour — but the relaxed goal-space it can lead to zero-valued Bayesian posteriors.

For example, in Table 4.5, the posterior probabilities of `(at package2 loc3)` and `(at package3 loc6)` after the first observation would always be zero, as the action observed has contributed nothing towards its achievement. This is because the first observation is aimed at achieving the non-mutex goal `(at package1 loc3)`. As each literal is considered individually in the relaxed goal-space the posterior probability for `(at package2 loc3)` and `(at package3 loc6)` would be zero. This is despite both literals still being valid goals. Therefore, the standard Bayesian update formula (Equation 3.3) outlined in Section 3.4 must be modified to consider the possibility of conjunctive goals in the relaxed goal-space.

Inspired by work in Information Retrieval [164], a *smoothing* parameter, λ, can be introduced into the Bayesian update formula to remove the chance of a posterior probability being zero. Equation 4.21 shows this modification, which uses the constant $\lambda \in (0 : 1)$ to smooth probability updates for goal G with a non-zero value for $W(G|O)$[11]. If O has not contributed towards achieving G, the second term ensures a non-zero probability is returned for the likelihood function.

Until now, the likelihood function of Bayes' theorem and *work* performed have been equivalent. However, from this point onwards these should be treated as separate concepts.

$$\mathrm{P}(O|G,\mathcal{G}) = \Big[\lambda \times W(G|O)\Big] + \Big[(1-\lambda) \times \frac{1}{|\mathcal{G}|}\Big] \qquad \text{where } G \in \mathcal{G} \tag{4.21}$$

This minor modification prevents any goal being eliminated from the goal-space due to a posterior of zero. The effect of the smoothing constant for $W(G|O) = 0$ is to spread the work attributed to O across all $G \in \mathcal{G}_i$. The degree to which this work is spread is dependent upon the value of λ. Low values will cause a smaller value to be returned for the likelihood function, which will result in smaller posterior increments or decrements, with the opposite being true of high values.

[11]A value of $\lambda = 1$ will result in the same behaviour as if an optimal heuristic and single literal goal is assumed, while $\lambda = 0$ would ignore all observed evidence.

High values of λ would emulate the original assumption of optimality in the agent and completeness in the goal-space, while low values would indicate that observations had little impact upon the goal being pursued. Note that if this value was different for each goal, the value of λ would equate to how relevant the observation was for each goal (or how "trustworthy" it is in a partially observable environment). For example, a high value of λ would mean that the observation is highly relevant to the posterior of a goal, while low values would indicate that it is irrelevant or perhaps untrustworthy. However, as stated, this work does not explore this possibility and instead uses a fixed value for λ.

Indeed, the value of λ used within IGRAPH is somewhat moot, as the algorithm used to construct hypotheses (page 79) will always output the same set of goals regardless of λ value used. This is because a constant $\lambda \in (0:1)$ value simply dampens the resulting posterior probability, regardless of the specific goal. For instance, using a value of $\lambda = 0.99$ on uniformly distributed goals $G_1, G_2 \in \mathcal{G}_1$ may produce posteriors of $\mathrm{P}(G_1) = 0.995$ and $\mathrm{P}(G_2) = 0.005$, while a value of $\lambda = 0.01$ results in $\mathrm{P}(G_1) = 0.505$ and $\mathrm{P}(G_2) = 0.495$. For all values of λ, goal G_1 will always be inserted into the resulting hypothesis over G_2. Therefore, while possibly interesting for future work, the λ value currently serves only to prevent zero-value posteriors.

Stability as an Indication of Goal Likelihood

One of the underlying assumptions given previously is that agents will always strive to retain any goals which are achieved prior to plan termination. That is, if the agent achieves one of their goal literals on the first observation of an n-step plan, they will try to keep this literal true over the remaining steps, such that it is never negated then re-added later.

This is simply the goal's *stability*, as defined previously in Section 4.3.2. This stability, $\Upsilon(G) \in (0:1]$, is the number of timesteps over which the goal has been true since its first achievement, and serves as a useful addition to the likelihood function, $\mathrm{P}(O|G)$.

$$\mathrm{P}(O|G,\mathcal{G}) = \Big[\lambda \times W(G|O) \times \Upsilon(G)\Big] + \Big[(1-\lambda) \times \frac{1}{|\mathcal{G}|}\Big] \text{ where } G \in \mathcal{G} \quad (4.22)$$

The inclusion of the goal stability can provide context to facts which rapidly fluctuate between true and negated, that would otherwise be added to hypotheses. For example, in the ZENOTRAVEL domain, planes must carry passengers between airports. Given that the same plane may visit the same airport multiple times within a plan, the stability of the goals describing the plane's location (such as `(at airport1 plane)`) will dwindle after the first visit. If the stability is not considered, the movement between airports can result in a uniform distribution across those airports the plan can fly to, which leads to a poor hypothesis. However, if the stability is included as shown in Equation 4.22, the resulting value for the likelihood function will be dampened such that it is lower than previously computed. By itself this is not of great importance, as the *posterior* probability will not change greatly if all facts considered in the Bayesian update exhibit this behaviour. However, recall that the *all false* goal which exists in every sub-goal-space will always have a stability of 1. Therefore, problems in which the standard positive literal members of a sub-goal-space have low stability, can potentially result in the *all false* goal having a high posterior due to a higher-valued likelihood function. This is the desired behaviour, as throughout observation the recogniser will begin to believe that low-stability goals are of no relevance in being included in hypotheses.

Of course, it is possible for facts which have low stability values at the end of recognition to actually be part of the true goal. These were termed *maintenance* goals in Section 3.6.2, where they were described as being a goal which is only actively pursued for part of the plan (after being negated). By incorporating the stability of the goal in the likelihood function, detection of these maintenance goals becomes more difficult as posterior probabilities will be lowered. However, the use of the goal stability does not *prevent* the detection of these goal types, merely that this detection is less likely.

4.3.7 Bounded Hypotheses in a Relaxed Goal-Space

Section 3.4.3 defined a method for estimating the number of observations remaining, $\varepsilon \in \{x \in \mathbb{Z}^+ | 1 \leq x \leq \tau_{max}\}$ as being equal to the number of steps required to achieve the goal in $\mathbb{G}$ with the highest associated probability (where τ_{max} is the distance from the current state to the furthest goal). In the relaxed model, these same principles apply, with the exception that the heuristic estimate may now be inaccurate.

The impact of this loss of accuracy is naturally that the number of estimated steps to achieve the same goal may change (or remain the same) between observations. For example, if $h(G) = 4$ after an observation, the successive estimation may be $h(G) \geq 4$. In turn, this will impact the number of *bounded hypotheses* produced.

After each observation, n bounded hypotheses can be produced as described in Section 3.4.4 (page 58), where $1 \leq n \leq \tau_{max}$. In the relaxed model these hypotheses are still computed in the same manner as previously, but the possibility of inaccurate values of n may cause extraneous hypotheses to be produced. While the method for extracting these hypotheses remains unchanged, it is no longer applied to the multivariate goal-space, but rather to each sub-goal-space in $\mathcal{G}^V$.

Bounded hypotheses are constructed in an identical manner to *intermediate* hypotheses, albeit with a smaller pool of goal literals to choose from. Whereas intermediate hypotheses can select any combination of goals across $\mathcal{G}^V$, bounded hypotheses are limited to those which have an estimate of $1 \leq h(G) \leq \tau_{max}$. If a bound, b, is placed on the hypothesis, the set of potential goals is constrained to those where $1 \leq h(G) \leq b$. In other words, only goals which can be achieved after the current state but prior-to or at the bound are considered valid candidates.

This subset of the goal-space is then used to construct a greedy hypothesis in the same manner as given in Algorithm 3. This approximates the behaviour of the complete model by allowing conjunctive bounded goals to be considered.

4.3.8 Library Integration

In the relaxed goal-space, only single literal facts are enumerated to enable tractable recognition. However, this does not disallow the presence of conjunctive goals in $\mathcal{G}$. Such goals can largely be treated in the same manner as single literal goals, making integration of an existing goal-library a simple operation.

By including conjunctive goals, the observer can normally extract more accurate heuristic estimates, and therefore improve upon recognition. The only feature of the complete model which would need to be modified would be related to the formation of sub-goal-spaces. Given that each sub-goal-space is formed from a mutex-set, this set would need to be modified to consider the new conjunctive goals. For instance, the mutex set $M = \{A, B\}$ would have to be extended to include goals which were mutex with relevant conjunctions, such as AC.

In reality, this modification could potentially be ignored. By leaving the original mutex sets/sub-goal-spaces in their standard form and instead considering each conjunction to only be mutex with its negation, the original algorithms used for relaxed recognition can be retained. However, this is not ideal, as it invalidates the benefits in considering the goal-space as a distribution over mutex-sets, and moves the model towards the original complete goal-space wherein all goals are mutex with one another.

Rather, the *greedy relaxed hypothesis* extraction algorithm (definition 18, page 78) would need to be modified to account for situations in which a conjunctive goal and mutex single literal goal are put forward by the basic *relaxed hypothesis* extraction algorithm (algorithm 3, page 79). For example, if conjunctive-goal AB has probability $\mathrm{P}(AB) = 0.7$ and the mutex goal BC also has $\mathrm{P}(BC) = 0.7$, the decision as to which goal (A, B, C, AB or BC) is included in the hypothesis must be determined by the greedy algorithm.

4.3.9 Goal Abandonment

The ability to detect an agent abandoning their goal mid-plan was discussed briefly in Section 3.6, where it was shown that this is implicitly catered for in the heuristic-based model.

Modelling of goal abandonment is retained in the relaxed model of recognition. As heuristic estimates are still used as the primary indicator of a goal being pursued, a goal being abandoned should still result in a decrease in the associated probability — although suboptimal heuristics may have an influence on the speed of detection. Abandonment is also unaffected by the introduction of the single literal goal-space, as abandoning individual goals can be treated in the same manner as conjunctive goals in the complete goal-space. However, there are also a number of other factors which will determine whether an abandoned goal is successfully detected.

- **Helpful Observations** — The most important consideration in detecting goal abandonment is naturally the number of observations which contribute towards the original goal, versus the successor goal. As the likelihood of a goal being abandoned is not computed separately, the switch between goals is treated as a suboptimal (or irrelevant) series of early observations in the overall plan. A high number of helpful observations for the first goal may cause the second goal to be undetected. This is largely determined by the *work* function used in the respective configuration of IGRAPH.

- **Work Function** — Naturally, the *maximum likelihood* work function, W_{ML}, is targeted at optimal plan traces where the agent does not abandon the goal. Therefore, configurations using this function are expected to perform poorly when detecting goal abandonment. W_{MLT} will perform identically to W_{ML} in cases where the observed plan is totally-ordered. The abandoned goal will only be found when W_{MLT} determines that the successor plan could have been started at an earlier time **and** the number of helpful observations for the latter goal exceeds that of the first, making it similarly unhelpful in detecting abandonment.

Finally, the W_{SA} function offers the simplest behaviour for detecting abandoned goals. As only the current observation is considered when determining whether an observation is helpful, the probability of the abandoned goal being the true goal will be exceeded once the successor goal has a higher number of helpful observations.

- **Abandonment Time** — The time at which a goal is abandoned will have a large impact on detection. Goals abandoned early or late in the initial plan may be easier to detect than those abandoned midway. In the former case, this is because there will have been minimal commitment to the goal, while in the latter it is often the case that the work put towards achieving the abandoned goal will also be useful in the context of the successor goal (leading to a natural progression from the first to second goals). Goals abandoned in the middle of a plan do not offer these properties, forcing detection to be one of counting helpful observations.
- **Stability** — While unlikely for most problems, it is possible that the goal which is abandoned may have been achieved and negated prior to abandonment. Alternatively, the successor goal may be possible only through achieving the former goal. In situations like this, the *stability* of the abandoned goal will be lowered, impacting upon the overall probability (as given in Equation 4.2).
- **Successor Goal** — As the previous point hints at, the agent's successor goal can dictate whether abandonment is viewed as simply a suboptimal plan achieving the successor, or as an altogether different plan. For a successor goal which is achieved through a similar set of plans to the predecessor goal, detection is simply a matter of noting that the predecessor goal has been achieved and that further actions have been observed. However, if the abandoned and successor goal have no relation (altogether different plans/areas of the goal-space), detection is determined by the criteria given above.

Each of these factors will have a role in determining whether the agent abandoning their goal is detected. In most cases, the total heuristic movement will be the primary factor in this process, although the work function and time of abandonment can also play a crucial role. Chapter 5.16 investigates how some of these elements affect abandonment detection.

4.3.10 Relaxed Model Conclusions

The relaxed model of goal recognition given in the preceding section enables tractable recognition to take place using a linear goal-space representation and suboptimal heuristics for both observer and agent. In this form, IGRAPH can be seen as a baseline implementation of the model given in Chapter 4.1, wherein observations are used purely as a means of determining the agent's goal.

Knowing that the agent will actively pursue their goal may be sufficient to perform accurate recognition, in that the agent's actions contain enough information to derive their goal. However, many problems have an underlying structure which makes certain literals more likely to be goals than others. At a basic level, if these goals could be detected, then the *uniform* initial probability distribution which has been assumed until now can be eliminated, and replaced with one which favours these highly probable goals.

4.4 Inferring Structure From Domain Analysis

Section 4.3 introduced several relaxations to the previous optimal and complete model given in Chapter 3, in order that goal recognition can become tractable for non-trivial problems, without prior knowledge of the domain. Yet, while goal recognition can be successfully performed using this new model, only information encountered during observation (i.e. the observed actions themselves) is used in inferring the agent's goal. Thus, any information which can be extracted from the domain at runtime is neglected, despite this potentially aiding in recognition.

IGRAPH introduces several techniques which are used to help "bootstrap" recognition — *fact partitioning*; *goal causality*, and *hierarchical goal likelihoods*. These are primarily targeted at the generation of an initial probability distribution, which can assist in producing more accurate hypotheses in the early stages of observation. *Domain analysis* is a common technique within the planning literature [16, 40, 51, 52, 53, 64, 65, 73, 84, 100, 138, 148, 149, 161], where it is used to assist in focusing search towards the goal. In contrast, domain analysis is all but unknown in recognition[12], as the availability of a plan-library negates any need for feature extraction. This is unfortunate, as the presence of a plan-library could only aid in the automatic derivation of domain knowledge.

4.4.1 Fact Partitioning

In any recognition task, a human expert can quickly derive the most probable goals and methods with which to achieve them. This can often extend to any human observer, who can draw on observations and their own previous knowledge to infer the agent's goal. In a propositional world such as that presented, these common goals can sometimes fall into *partitions*, such that members of one partition are more likely to be selected by a human observer as being in (or not in) the goal set.

In IGRAPH, partitions are created for a subset of the goals contained within $\mathcal{G}^V$, prior to observation beginning. Any information extracted can then be used in the generation of the initial probability distribution, or during recognition. Specifically, three non-overlapping subsets of the goal-space are extracted as follows, where $g \in \mathcal{G}^V$ and π is the planning problem as defined on page 34.

Definition 23. Fact Partitions:

1. **Strictly Activating** — A fact g, is strictly activating if $g \in \pi(I)$ and $\forall a \in \pi(A)$, $g \notin a_{add} \cup a_{del}$. Further, $\exists a \in \pi(A)$, $g \in a_{pre}$, where $\pi(A)$ is the set of *grounded actions*.

2. **Unstable Activating** — Any fact $g \in \pi(I)$ is unstable activating if $\forall a \in \pi(A)$, $g \notin a_{add}$ and $\exists a \in \pi(A)$, $g \in a_{pre}$ and $\exists a \in \pi(A)$, $g \in a_{del}$.

3. **Strictly Terminal** — If $\exists a \in \pi(A)$, $g \in a_{add}$ and $\forall a \in \pi(A)$, $g \notin a_{pre}, g \notin a_{del}$.

Simplified, the partitions behave in the following manner. *Strictly terminal* (ST) facts are those which do not appear as a precondition to any action, and once added, cannot be deleted. Of all partitions, these are the most likely goal candidates, as their achievement serves no other purpose (i.e. there is no causal link between these and other goals). It is possible that an agent would wish to *prevent* a ST goal becoming true, as this may represent an unwanted world state. However, without observed evidence this is impossible to determine, but once

[12]With the recent exception of Keren *et al.* [99], although this is performed offline and involves the generation of an optimal plan-space.

this is available it should be clear whether the goal is being pursued or avoided. Therefore, it is assumed that ST facts are indeed more likely to be goal candidates than not.

Strictly activating (SA) facts are the opposite of ST facts. While members of the ST partition cannot be deleted once added, SA facts can never be deleted *or* added. This means that all members of this partition must be true in the initial state, such that $\forall f \in SA \in \pi(I)$. In practice, these facts can be eliminated from the goal-space and play no role in recognition, as they fundamentally cannot form part of the agent's goal. Instead, they are used to provide problem structure, such as indicating that two locations are connected via a road.

Finally, the *unstable activating* (UA) partition contains facts which once deleted, cannot be re-achieved. Like the *strictly activating* set, any member of this set must be true in the initial state, but can be subsequently deleted. These are often associated with *dead-ends*. For instance, deletion of fact X may prevent execution of actions which achieve the goal (or part of the causal-link towards the goal). Alternatively, these can form a *fact resource*, which is consumed in achieving the goal. Once deleted by an observation, these too can be removed from the goal-space.

By themselves, these partitions can provide a non-expert human operator with additional information, but they can also offer assistance in lowering the size of the goal-space or in constructing hypotheses. In particular, if a member of the ST set is achieved, it will *always* be added to any future hypothesis. This is in keeping with the assumption that ST facts are goals which are desired by the agent to be achieved, rather than avoided.

The above definitions allow certain members of the goal-space to be highlighted as being of interest to the observer for various reasons. This is done by examining the relationship of each goal with the actions that require, add or delete it. While only three partitions are defined, the concept of using this relationship between goals and actions can be extended to all members of the goal-space. This amounts to computing the probability of a fact being added to the domain *as a goal*, versus it being added in order to be achieved by another action, or threatened with deletion by other actions. This is referred to as the *risk* associated with a goal.

4.4.2 Goal Likelihood as Risk of Negation

In most goal recognition problems, certain goals will have a higher probability of being the agent's true goal than others. Fact partitioning provides a means of extracting a (possibly small, or even empty) subset of the domain which exhibit strict properties that can be exploited. However, this does not assist in the generation of a non-uniform initial probability distribution across $\mathcal{G}$. Members of each partition could be assigned manual probabilities, but this would leave a large subset of the goal-space untouched, and these weights would not be domain-independent.

In a probabilistic model such as that presented, each goal must have a *prior probability* assigned before recognition begins. Traditionally, these probabilities are determined by a domain-expert [60, 70, 97] or by performing statistical analysis on a set of the agent's previous plans [3, 18]. For the latter, this can mean observing, or at least having access to, a large number of plans generated by the agent, while the former requires the time, availability and knowledge of someone versed in the problem domain.

Both of these approaches to assigning prior probabilities are common to the recognition literature, yet they are both unreasonable to assume in widespread application. Moreover, they represent a bottleneck in the deployment of a recognition system. As such, there should always be a *domain-independent* method of estimating prior probabilities based purely upon domain analysis when no further information is available. The naïve method of achieving this is to assign a *uniform probability distribution* across each sub-goal-space as follows[13].

$$\mathrm{P}(G) = \frac{1}{|\mathbb{G}|}, \forall G \in \mathbb{G} \tag{4.23}$$

While this does provide a basic means of assigning weight, it offers little insight into the most probable goal before recognition begins. IGRAPH assigns probabilities to facts by making the assumption that the subject will try to maintain a goal literal once achieved. Given that the observed agent will achieve its goal through a series of causal-links, the notion of *goal causality* is introduced to provide an indicator of which facts are most likely to be goals. This can be defined as follows, and expressed in Equation 4.24.

Definition 24. Goal Causality
Given a goal-space $\mathbb{G}$, and action set $\pi(A)$, the most probable goals $G \in \mathbb{G}$ are those which interact the least with members of $\pi(A)$, once G has been achieved. This can be expressed as a *causality value*, $\mathcal{C}(G) \in [0 : 1]$.

[13]The original notation for the goal-space, $\mathbb{G}$, is used here as the concept of goal-causality extends to both the complete and relaxed models.

$$\mathcal{C}(G) = \frac{|A_{achievers}(G)|}{|A_{achievers}(G)| + |A_{deleters}(G)| + |A_{requirers}(G)|} \tag{4.24}$$

Each set, $A_{achievers}(G), A_{deleters}(G), A_{requirers}(G) \in \pi(A)$ corresponds to a subset of the action set which achieve, delete or require G as a precondition respectively. This equation allows a probability $\mathrm{P}(G) \in [0 : 1]$, to be assigned to each fact, based upon its likelihood of being deleted or being part of an intermediate causal-link in the plan.

In the case of *all false* goals, the assumption set out in Chapter 3 stating that the agent's goal will be a positive literal is retained. However, the notion of goal causality does not easily extend to *all false* literals. Therefore, $G_\emptyset$ is initialised with the same value as the *uniform initial distribution* would produce.

Once initial probabilities have been determined, they can be normalised across each sub-goal-space to achieve a non-uniform initial probability distribution. Note that is often not possible to get the probability distribution across the *entire* multivariate goal-space (where each goal appears only once), as goals can appear in more than one sub-goal-space, where probabilities can vary greatly.

Detecting a fact's *goal causality* allows the creation of a domain-independent distribution which is weighted towards those members which appear to be the target of planning rather than the requirement for planning. However, further analysis can influence the initial probabilities towards those members of the goal-space which the underlying domain structure indicates will form part of the final goal.

4.4.3 Utilising the Causal Hierarchy

Recall that the causal graph, CG, as defined in Section 4.2.1 represents how objects interact with one another in a given problem. While the majority of these interactions will be bidirectional, certain problems and variables have unidirectional influences, such that an object will only influence *or* be influenced by another. If an object can only be influenced by others, then it is classed as a *leaf variable*, while if it exclusively influences other objects, it is a *root variable.*

Definition 25. Leaf and Root Variables
A node $v \in CG$ in the causal graph CG is a *leaf variable* if $|v_{in}| > 0$ and $|v_{out}| = 0$. Conversely, v is a *root variable* if $|v_{in}| = 0$ and $|v_{out}| > 0$.

Intuitively, leaf variables are those which are most likely to contain goal literals in their associated domain transition graph, while roots are the least likely to appear in the goal. For example, in the Driverlog domain [117], packages are leaf variables — they exist only to be moved from one location to another. On the other hand, in the Rovers domain, rovers themselves are roots — they influence the derivation of all facts in the domain (aside from those in their associated DTG).

Based on the rationale that the leaves of the causal graph are the most likely to appear in goal literals, and that the roots are therefore the least likely, the causal graph can be converted into a *hierarchical causal graph*, CG^H. This involves assigning a *layer*, L_v, to each variable $v \in CG$. Algorithm 11 outlines the procedure for determining each variable's layer, while Algorithm 12 demonstrates how to extract the layer of an individual goal literal. In the event that the causal graph has neither roots nor leaves, hierarchical analysis cannot be performed and all variables are said to lie on the same layer, $CG^H(v) = 1, \forall v \in CG$.

With a layer assigned to each variable as in Figure 4.11b, the initial probability distribution can be augmented to incorporate this as in Equation 4.25. This is done by obtaining the *minimum* layer at which each goal in $\mathcal{G}^V$ lies, and adjusting the causality value $\mathcal{C}(G)$. Using the minimum layer causes the fact to receive the highest possible initial probability.

$$\mathrm{P}(G) = \frac{\mathcal{C}(G)}{\arg\min L_v(G)} \forall v \in CG^H \tag{4.25}$$

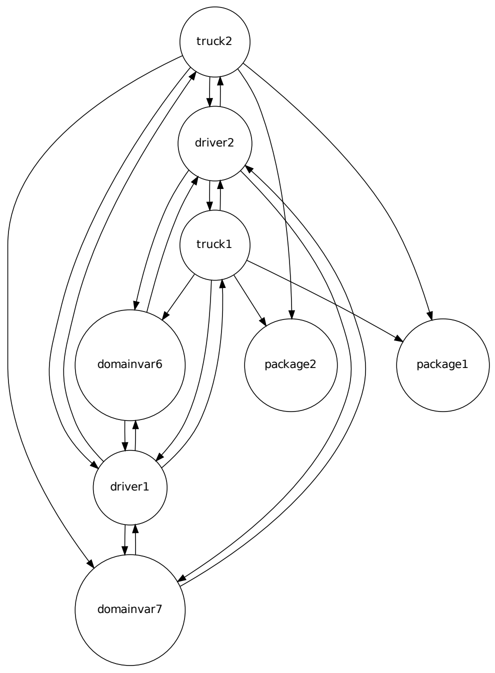
4.4.4 Domain Analysis Conclusions

Analysis of the domain in which recognition is being performed is in line with the assumptions set out previously. Here, goals are assumed to remain true once achieved, with the *causality* of the goal being used as an indicator of how likely it is to form part of the true goal. This value is then weighted according to the layer of the causal graph which the goal appears in, as it is believed that goals which appear as leaves/on lower levels are more likely to be a part of the agent's goal than those at higher levels. Finally, fact partitions are used at runtime as a means of appending facts to hypotheses or pruning areas of the goal-space.

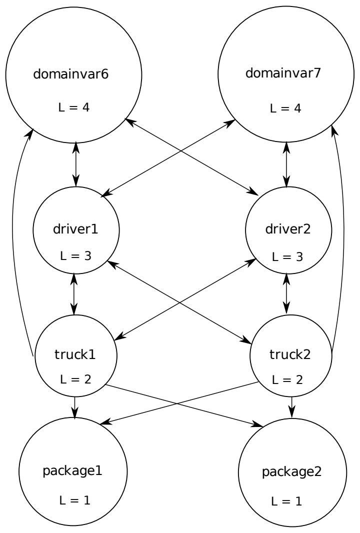
Work by Keren *et al* [99] uses planning to assist in the design of PR/GR problems, in order that recognition becomes simpler. While not required, it may be that the design of a domain which IGRAPH is targeted at can be modified, so as to maximise the benefits of the above feature extraction processes. For

Algorithm 11 createHierarchy(CG)

```
$CG^H := \langle DTGs, 1\rangle$ {Initialise all DTGs to layer 1}
$closed := \{\}$
**if** $CG_{roots} = CG_{leaves} = 0$ **then**
    **return**
**end if**
**if** $CG_{leaves} > 0$ **then**
    $Q := \{CG_{leaves}\}$ {Initialise queue with leaves}
    **while** $|Q| > 0$ **do**
        $v := poll(Q)$
        $prev := CG^H(v)$
        $add(closed, v)$
        **for all** $out \in v_{out}$ **do**
            **if** $prev \in closed$ **then**
                continue
            **end if**
            $curr := prev + 1$
            $add(Q, out)$
            $add(CG^H(out), curr)$
        **end for**
    **end while**
**else**
    $Q := \{CG_{roots}\}$ {Initialise queue with roots}
    $min := |CG|$ {Set minimum layer to size of CG}
    **while** $|Q| > 0$ **do**
        $v := poll(Q)$
        $prev := CG^H(v)$
        $add(closed, v)$
        **for all** $out \in v_{in}$ **do**
            **if** $prev \in closed$ **then**
                continue
            **end if**
            $curr := prev - 1$
            $add(Q, in)$
            $add(CG^H(in), curr)$
            **if** $curr < min$ **then**
                $min := curr$
            **end if**
        **end for**
    **end while**
    {Need to modify layers such that $\exists v \in CG^H, L_v = 1$}
    **for all** $v \in CG^H$ **do**
        $L_{old} := CG^H(v)$
        $L_{new} := L_{old} - CG^H(v) + 1$ {Add 1 to ensure no variable is $L_v = 0$}
        $CG^H(v) := L_{new}$
    **end for**
**end if**
```



(a) The causal graph as output by SAS$^+$ translation.



(b) The equivalent hierarchical causal graph in which every object has been assigned a layer, L.

Figure 4.11: The causal graph for a DRIVERLOG problem before and after detection of each variable's layer. Note that variables "domainvar6" and "domainvar7" are generated by SAS$^+$ translation to model whether a truck is empty or not (the corresponding "truck occupied" literals appears in a *driver's* mutex set).

Algorithm 12 getGoalLayers(CG)

```
$minLayers :=< \mathcal{G}, 1 >$ {Initialise all goals to layer 1}
if $CG_{roots} = CG_{leaves} = 0$ then
  return
end if
for all $dtg \in DTGs$ do
  $L_{prev} := minLayers(dtg)$
  for all $G \in dtg$ do
    $L_{curr} := minLayers(G)$
    if $L_{curr} < L_{prev}$ then
      $add(minLayers(G), L_{curr})$
    end if
  end for
end for
```

example, addition of ST facts as goals or ensuring that the causal graph contains roots/leaves.

4.5 Chapter Summary

This chapter has introduced multiple relaxations to the complete and optimal model presented in Chapter 3. This relaxed model retains all of the desired features discussed in Section 2.4, which are repeated here for clarity.

- **Removal of Plan Library Dependence** — Plan libraries are not required for recognition to be performed. Instead, the goal-space is generated by analysing the possible set of goals. This set is an over-estimation of the true goal-space, as it can contain unreachable or invalid facts. However, all valid facts should be present. The agent's goal should always be present within the relaxed goal-space, although generation of this as a conjunctive hypothesis may not be possible due to the incomplete detection of mutually-exclusive facts in the domain. An approximation of this is often possible, but may be invalid as there may be mutually-exclusive facts present which are not simple to detect at runtime.

- **Automated Initial Distribution** — Section 4.4 demonstrated several ways in which domain analysis could be used to extract further information from the domain which may aid in recognition. In particular, the automatic generation of an initial probability distribution across each sub-goal-space $\mathcal{G}_i \in \mathcal{G}^V$ is performed by computing the *causality value* of each goal and its corresponding, minimal layer in the *hierarchical causal graph.*

- **Plan/Goal Abandonment** — Detecting and adjusting to an agent abandoning their goal was expanded upon in Section 4.3.9. Chapter 5 will provide a demonstration of this ability.

- **Standardisation** — IGRAPH receives its input from a common formalism. PDDL [55] is used to define a domain and problem in a format similar to that of STRIPS [49], from which a *grounded* problem is generated. These input files can also be converted into a SAS$^+$ form [9] using an existing translator [78]. By using a standardised input, problems can be constructed more rapidly, and existing domains, problems and solutions can be used in evaluation.

With the relaxed model finalised, the next chapter will present the results of evaluating this model against a series of domains taken from various International Planning Competitions. This will cover the classic goal recognition behaviour, as well as more novel aspects of the model. In addition to this, features such as the source of plan, heuristic used in generation of this, and plan length will be explored.

Chapter 5

Evaluation

This chapter presents results of several evaluation schema on IGRAPH. Rather than only reporting the accuracy of hypotheses with regard to the agent's final goal, the heuristic-based GR model is explored in further detail. This has been done to demonstrate several features which may not be present in previous models.

5.1 Overview

The chapter begins with an overview of the evaluation techniques applied (Section 5.2), and describes how these compare with previous evaluation metrics in the recognition literature (Section 5.3). Section 5.4 discusses how implementation of a heuristic-based recogniser can be achieved using an existing planner. The typical test setup used during evaluation along with any specific parameter values is given in Section 5.5. The expected output of the heuristic-based model is briefly discussed in Section 5.6, along with the IPC domains[1] and plan sources used in this chapter (Sections 5.7). Section 5.8 then details the source of plans used during evaluation of the International Planning Competition domains and their suitability in recognition, while Section 5.9 provides some context on the size of the respective goal and action-spaces.

Section 5.10 evaluates the accuracy of *intermediate* hypotheses within IGRAPH, both in the context of the final goal and the final state. Evaluation of final hypotheses themselves is covered in Section 5.11. The accuracy of the recogniser in

[1]See Appendix D for a list of domain descriptions.

predicting goals before and after their achievement is given in Section 5.12. Sections 5.13 and 5.14 respectively describe the accuracy of the estimated number of remaining steps and the bounded hypotheses derived from this figure.

The impact of generating a non-uniform initial distribution on the accuracy of hypotheses produced throughout observation is provided in Section 5.15. Detection of goal abandonment in under various conditions in the heuristic-based model is explored in Section 5.16.

Section 5.17 provides analysis on whether recognition accuracy is improved by using plans generated with a domain-dependent heuristic, while Section 5.18 looks at whether there is a bias in results when the observer and agent use the same heuristic.

Finally, Section 5.19 provides possible explanations for some of the unexpected behaviour exhibited by IGRAPH, before 5.20 concludes the evaluation and highlights the observed results of the heuristic-based model of recognition.

5.2 Evaluation Techniques

In evaluating any system, it is often helpful to have a single metric which can be used to demonstrate that system A is better than system B on problem X. Unfortunately, the plan and goal recognition community has yet to settle upon a common metric for goal, plan, intent or activity recognition, despite this being an ongoing topic of discussion [18, 35, 71, 132]. This is in part due to the decentralised nature of the field, and in part to the absence of a common input/output formalism. As a comparison, the planning community has had both a standard language [47, 55, 67, 120] and evaluation schema [117] for over a decade, which has aided in bringing the community together and fostering competitive research.

However, this is not to say that *every* recognition-related work is evaluated using a unique metric. In practice, *precision and recall* (or extremely close variants of this) has emerged as a semi-standard metric for goal recognition, having been applied in several GR papers [3, 18, 20, 109, 122]. This is also a useful metric for the heuristic-based GR model, as the following section shall demonstrate.

5.2.1 Precision and Recall

The *precision and recall* (P+R) metric is often associated with database document retrieval, where it is used to evaluate the accuracy of the results returned by a search algorithm against the necessary results. While referred to as a singular metric, precision and recall are in fact two distinct equations, which can be used in conjunction to demonstrate the generalisation of a system versus its accuracy.

In the context of goal recognition, P+R can be used to analyse how complete a hypothesis is, versus how concise this is. That is, how many of the true goal literals $G^* = \{G_1...G_n\}$ are contained in the hypothesis (recall score), versus how many of the goals contained in the hypothesis Hyp are required (precision score). Formally, precision and recall are defined as follows.

$$Prec(Hyp) = \frac{|G^* \cap Hyp|}{|Hyp|} \in [0:1] \tag{5.1}$$

$$Rec(Hyp) = \frac{|G^* \cap Hyp|}{|G^*|} \in [0:1] \tag{5.2}$$

Thus, the precision of a hypothesis represents the number of goals present in the hypothesis which are necessary, while recall is number of goals in the hypothesis which are actually correct (part of G^*). In the case of precision, a score of $Prec(Hyp) = 1$ means that all literals in the hypothesis are a member of G^*, while a score of $Rec(Hyp) = 1$ means that all true goals were contained in the hypothesis.

As an example, consider hypothesis $Hyp = \{A, B, C\}$ against the true goal $G^* = \{A, B, X, Z\}$. Here, the precision and recall scores would be as follows.

$$\begin{aligned} Prec(Hyp) &= \frac{|G^* \cap Hyp|}{|Hyp|} \\ &= \frac{|\{A,B,X,Z\} \cap \{A,B,C\}|}{|\{A,B,C\}|} \\ &= \frac{|\{A,B\}|}{|\{A,B,C\}|} \\ &= \frac{2}{3} \\ &= 0.66 \end{aligned} \qquad \begin{aligned} Rec(Hyp) &= \frac{|G^* \cap Hyp|}{|G^*|} \\ &= \frac{|\{A,B,X,Z\} \cap \{A,B,C\}|}{|\{A,B,X,Z\}|} \\ &= \frac{|\{A,B\}|}{|\{A,B,X,Z\}|} \\ &= \frac{2}{4} \\ &= 0.5 \end{aligned}$$

F$_1$ Score

As has already been stated, it can be useful to have a *single* metric against which to compare results. While precision and recall provide a means of quantitatively evaluating a hypothesis, the output requires a qualitative examination. For instance, if there are two hypothesis Hyp_1 and Hyp_2 with scores of

$Prec(Hyp_1) = 0.6, Rec(Hyp_1) = 0.8$ and $Prec(Hyp_2) = 0.9, Rec(Hyp_2) = 0.5$, it is unclear as to which is the better result.

One solution to this is to compute the F_1 score [157], which returns a single value $F_1 \in [0 : 1]$, that is intended to provide a trade-off between precision and accuracy.

$$F_1(Hyp) = 2 \cdot \frac{Prec(Hyp) \cdot Rec(Hyp)}{Prec(Hyp) + Rec(Hyp)} \tag{5.3}$$

The F_1 score is in fact a specialisation of the general F_β score, in which $\beta = 1$. Higher values of $\beta > 1$ would be used to indicate that the recall score is of higher importance than the precision, while values of $0 < \beta < 1$ would indicate that precision has greater precedence.

$$F_\beta(Hyp) = \begin{cases} (1 + \beta^2) \cdot \dfrac{Prec(Hyp) \cdot Rec(Hyp)}{\beta^2 \cdot Prec(Hyp) + Rec(Hyp)} & \text{where } \beta > 0 \end{cases} \tag{5.4}$$

In the previous example, the F_1 scores for Hyp_1 and Hyp_2 would be $F_1(Hyp_1) = 0.69$ and $F_1(Hyp_2) = 0.64$, indicating that Hyp_1 is the superior hypothesis. However, it is arguable that in goal recognition, the *recall* score is of higher importance. That is, the true goal being contained within Hyp is of more importance than the size of Hyp. This is especially prudent when it is considered that Hyp will approximate a set of mutually-exclusive goals — meaning as the size of Hyp increases, it moves towards being a *state hypothesis*, $1 \leq |Hyp| \leq |\pi(M)|$. Of course, in the context of IGRAPH, these are *relaxed hypotheses* where the goals proposed cannot guarantee to be free of mutually-exclusive facts.

Given that recall is often more important in goal recognition, deriving the F_2 score may arguably be a better choice for evaluation. However, in order to prevent further evaluation-schema fragmentation within the recognition community, the F_1 score is retained.

5.3 Evaluation in Previous Recognition Literature

As referred to above, no common or *de facto* evaluation metric has emerged from the literature despite anticipation that this would occur [35, 132]. For many recognition systems, *time* is the primary metric of evaluation, with shorter processing times being preferable [6, 21, 60, 63, 89, 113, 114, 146]. Runtime results for

IGRAPH are reported in Appendix C, although computational resource usage is not viewed as critical to this work.

Precision and recall have been used by others in recognition, in particular Blaylock uses P+R [19, 21], although this definition is not the same as that given above. Others which use Blaylock's plan corpora also use the same metrics [3, 4, 122]. Additionally, Blaylock defines the *convergence* point of the recogniser — the time at which the correct goal is first hypothesised. This metric is also technically suitable for IGRAPH, but is omitted as the heuristic-based model does not lend itself to convergence upon a single hypothesis. Rather, the hypothesis is made of multiple individual facts which can be added or removed after an observation, making the concept of a "convergence point" unwieldy.

Other more bespoke metrics include prediction of the next action and agent location [1]; *impact*, which seeks to quantify how much work/effort has been saved by recognising the goal/plan and automating the process (akin to the *convergence* point defined above) [110, 112], and *efficiency coefficients* [31] which measure relative accuracy versus processor time. Others simply use the ratio of tests which correctly identified the goal [89].

Plan Library Resources

Similar to the lack of a common evaluation metric emerging from the community, a suite of plan libraries on which to test models has also not appeared despite the expectation that more connectivity and dissemination of research would achieve this [35]. While the use of libraries remains widespread, these structures are often extremely domain-specific [107, 108, 118, 154, 162][2], and do not generalise across the field of recognition.

The most commonly cited libraries used are Lesh's UNIX domain [89, 111, 112], and Blaylock's LINUX and MONROE domains [3, 4, 21, 144, 155], with LINUX being a modernised version of UNIX. Unfortunately, all of these domains are encoded in a hierarchical fashion, and no suitable translator could be found to encode them in PDDL. Finally, Ramirez and Geffner make use of PDDL and variants in their work [145, 146, 147], but in a largely partially-observable context, which IGRAPH does not support.

[2]Example datasets taken from `http://www.planrec.org`.

5.4 Development of a Recognition System Using an Existing Planner

IGRAPH has been implemented in Java, and builds upon the JavaFF planner [44], which has been extended to allow parsing of non-STRIPS domains[3]. JavaFF was chosen as the baseline planner due to a familiarity with the underlying structure and a readily available source of the FF heuristic. Yet, despite ostensibly supporting many of the features required by IGRAPH, approximately half of all files required alteration to achieve compatibility with the relaxed model. Beyond this, SAS$^+$ planning and STRIPS scheduling have been implemented, along with code required to perform the probabilistic reasoning required by the relaxed goal recognition model.

Specifically (with any alterations ignored), IGRAPH extensively uses the grounded and ungrounded PDDL type hierarchy and associated parser present in JavaFF. The planning and relaxed planning graph structures are used, as well as the FF-style relaxed plan extraction process. This allows both the h_{ff} and h_{max} heuristics to be used[4] without any additional modification. However, the h_{cea} heuristic and associated code base do not feature in JavaFF, and, along with the threading algorithm and supporting structures must be implemented separately.

When development of IGRAPH began, few existing planners had intuitive or supported code bases. However, this is no longer the case, with frameworks such as Fast Downward [5] offering almost all of the required features for heuristic-based goal or plan recognition. Indeed, one of the major advantages of the heuristic-based recognition model is that such libraries can be used as a code base. As much of the model presented in Chapter 4 uses existing planning technologies (heuristic, states, facts, goals etc.), these open source implementations should be usable for recognition with only minor additions. Were IGRAPH to be reimplemented in this framework, development time could be cut from months to days as only code related to the recognition process itself would be needed. As an example, Ramírez and Geffner make use of this framework in their 2010 work

[3]Specifically, in addition to the `:strips` and `:typing` PDDL tags, the modified JavaFF supports `:adl`, `:equality`, `negative-preconditions`, `quantified-preconditions`, `existential-preconditions` and `universal-preconditions` tags.

[4]The value for h_{max} can be derived from the relaxed planning graph, as being equal to the layer at which all required facts first appear.

[5]`http://www.fast-downward.com`

[146], which results in only a few additional files being required[6].

Integration of planner and recogniser can be even further expanded upon by making use of the recogniser's output in the planning process. For instance, in multi-agent planning, individual agents can use hypotheses generated by a recogniser to better plan opportunistically or to inform their own estimation of the world. Here only the pre-planning stage need be modified to account for hypotheses, as the existing probabilistic planning algorithm does not need to be changed, which again minimises any duplication of effort.

5.5 Test Setup

Test results have been collected on an 8-core machine running Ubuntu 11.10 64-bit, on an Intel Core i7 processor with a maximum 12GB of memory[7], although only approximately 2GB is ever in-use. Version 1.7 of the Java Virtual Machine has been used on the 64-bit OpenJDK. All tests were given as long as necessary to complete. Runtime results for processor time; wall clock time and memory consumed are available in Appendix C.

5.5.1 Standard Test Parameters

As there are several parameters present at various points in the IGRAPH recognition pipeline, it is helpful to make these values explicit, as many remain unchanged during 'standard' tests. As a rule-of-thumb, these parameters are as follows.

1. Unless otherwise stated, all IPC-related tests have been executed using the first 15 problems of the propositional variant of each associated domain. This figure has been selected as problems beyond this are generally intractable for IGRAPH, due to the exponential increase in runtimes due to the complexity of the problem. While not all domains exhibit this behaviour, retaining a standard problem-set size across domains is helpful in interpreting results. Runtimes themselves are reported in Appendix C,

[6]Code available at `https://sites.google.com/site/prasplanning/file-cabinet`

[7]As an interesting technical sidenote, the test-build of IGRAPH is launched through an external script, rather than directly calling Java. This is due to the fact IGRAPH must call the SAS^+ translation scripts [78] as a separate process. This causes Java to call `fork()` on the underlying operating system, which creates a new memory block of the same size as the JVM itself and assigns it to the external process. Thus, the maximum memory available to IGRAPH itself would be half the physical memory available. For this reason the SAS^+ translation is performed prior to calling IGRAPH, which simply assumes that the relevant SAS^+ files are correct for the current problem.

however for now it is sufficient to state that the runtime of IGRAPH is dominated by the complexity of the heuristic used and the number of goals in the goal-space, leading to the exponential behaviour mentioned.

2. A further detail regarding runtimes is related to the generation of *bounded hypotheses*. Recall that the number of bounded hypotheses generated after each observation is equal to the estimated number of steps remaining (Section 3.5). In certain domains such as Rovers and using the context-enhanced additive heuristic, the number of estimated steps remaining can be in the tens-of-thousands[8]. Therefore, while the number of estimated remaining observations ε is calculated, only $|P| - |O|$ hypotheses are produced, where $|P|$ is the true plan length and $|O|$ is the number of observations seen so far. That is, if there are n steps remaining in the plan, only $1 \leq \varepsilon \leq \tau_{max} n \leq \tau$ bounded hypotheses will be produced. While this naturally means that the runtime of IGRAPH cannot be accurately reported when bounded hypotheses are being produced, the actual time required for these is trivial and therefore is included in the runtimes given in Appendix C.

3. The use of domain analysis as described in the previous chapter should also be assumed to have been carried out prior to observation beginning. This entails solely of creating an initial probability distribution and performing a reachability analysis. The impact of not performing domain analysis and using a uniform initial distribution is explored in Section 5.15.

4. Regarding the use of λ in the Bayesian posterior probability update, a value of $\lambda = 0.8$ has been used across all tests. This reflects the assumption that the agent will always strive to achieve a minimal-length plan, but that they may do this using suboptimal heuristics. However, as Section 4.3.6 detailed, the exact value of this variable is irrelevant as it has no impact on the output of the planner (provided that $0 < \lambda < 1$). Therefore, while $\lambda = 0.8$ is a reasonable value to apply, varying this value does not change the hypotheses generated.

[8]For Rovers this is caused by the additive nature of the causal-graph heuristic, and the fact that there are often appears to be tens of non-mutually-exclusive *strictly terminal* goals being pursued at any time, when in reality only a single goal is being pursued. This causes the algorithm for estimating the number of remaining steps to output the sum of achieving all of these literals, which can be equal to thousands of actions, as positive interactions between these goals is not accounted for.

5.6 Expected Output

The formal model outlined in Chapter 3 stated that by having an optimal heuristic, the observer can always accurately infer the number of actions required to achieve each goal in $\mathbb{G}$. Given a rational agent, goal recognition becomes a matter of eliminating those goals whose estimate has not reduced after each observation. This was then relaxed to allow for suboptimal agents, which moved the model to include Bayesian probabilistic inference.

Chapter 4 stated that this view can be extended to include domains in which the observer has a suboptimal heuristic. However, even in this relaxed form, the central concept of an agent moving through the goal-space to achieve their final goal is retained, albeit without the absolute knowledge provided by an optimal heuristic. It is therefore expected that the result of testing will show *a heuristic whose estimates approach h^+ will provide more accurate hypotheses than one which is known to provide poorer estimates*, but only if the agent being observed demonstrates bounded rationality. As this final point is intractable to compute (determining whether a plan is optimal for all IPC solutions), the best known solution is assumed to approximate this.

Of the three heuristics implemented by IGRAPH (Section 4.3.3), h_{max} is expected to provide the poorest results, as it is known to be extremely uninformative under standard planning conditions [27]. Conversely, h_{ff} and h_{cea} are expected to perform far better, as they have been shown to be largely informative in the respective planners which initially implemented them [79, 88].

In addition to the standard test parameters presented above, most test problems have been evaluated using all *configurations* of heuristics and *work* functions (W_{ML}, W_{MLT} and W_{SA}, see Section 4.3.5). This leads to 9 different configurations for each test.

5.7 Evaluation of International Planning Competition Domains

The *International Planning Competition* (IPC) is a (roughly) bi-annual competition held at the International Conference on Automated Planning and Scheduling (ICAPS)[9] since 1998. Entrants compete in several competition tracks, aiming to produce the most "optimal" plans for the problem — where "optimal" can mean

[9]ICAPS was not formed until 2003. Prior to this the IPC was part of the AIPS conference, from which ICAPS was partly formed.

plan length, resources used or time taken.

While this competition is a useful source of heuristics for IGRAPH, it is the test domains and solutions produced which offer the most interest for evaluation. In particular, all IPC domains have been formalised in PDDL, which allows IGRAPH to use all of these in performing evaluation. With this said, evaluation is restricted to domains written in PDDL 2.1 level 1 [55], which is approximately equivalent to STRIPS [49] and ADL [134], with ADL domains being compiled into the equivalent STRIPS representation immediately after parsing.

In some cases, the translation of ADL to STRIPS-style representations will generate many more actions than would otherwise exist, which is expected to lead to slower execution times. This primarily impacts the computation of the h_{max} and h_{ff} heuristics, where the higher branching factor will mean more calculations are necessary. Computation of the h_{cea} heuristic is unaffected by this, as ADL is automatically compiled into additional "stub" facts which are then treated as standard preconditions/effects in SAS$^+$.

Historically, these languages have been associated purely with the non-metric tracks of the competition, in which the task is to find the shortest valid plan for the given problem. More recently, this has split into *optimal* and *satisficing* branches, in which competitors must find the shortest plan, or *any* plan respectively.

The domains selected for evaluation are taken from IPC3 and IPC5. The IPC3 domains include Depots, Driverlog, Rovers[10], Satellite and Zenotravel, while the IPC5 domains are Openstacks, Storage and Trucks. Detailed descriptions of these domains and their respective interesting features can be found in Appendix D. These domains have been selected as they conform to the PDDL 2.1 level 1 standard mentioned previously. Other domains within these competitions and others either use an unsupported features of ADL (IPC4), or include new PDDL syntax (IPC6, IPC7). Domains from IPC3 and IPC5 which have been excluded from testing have been done so because of bugs in the third-party SAS$^+$ translator.

5.7.1 Applicability of IPC Domains in Recognition

The IPC has been selected as the best source of domains for evaluation for several reasons. First, it offers a standardised set of problems which adhere to a strict description language, for which parsers are widely available. Second, the

[10]Note that the Rovers domain appeared in both IPC3 and IPC5. The IPC3 variant is used exclusively in evaluating IGRAPH.

domains used demonstrate several interesting characteristics which are common to the recognition problem. For instance, transportation of resources is present to a certain extent in all domains, while strictly terminal goals exist in Rovers, Satellite and Openstacks. All domains feature disjoint goals which can often be achieved using non-overlapping subplans that can, in turn, be executed in parallel. Goals may be achieved in any order or linearly, and in some cases may already be true in the initial state (where this means that they remain true throughout execution *or* must be negated then re-achieved later). The actual meaning of the domains (moving packages, observing locations from space etc.) are somewhat irrelevant, as it is the underlying structure and detection of movement through the goal-space which is of interest. Finally, the wide availability of the IPC domains used (and others) means that future researchers can use these in benchmarking their own recognisers against the results produced by IGRAPH.

One potential drawback to using IPC domains and problems is that the true goal often exists at the edge of the state and goal-spaces. For example, in some Driverlog problems, packages are often moved to the location furthest from their start point. Locating goals at a distance of $h(G) = \tau$ (particularly single literal goals) may lead to early observations having a higher impact than if the goal were located in the middle of the goal-space. This impact is dependent upon whether the heuristic is accurate in lowering the estimate during these initial observations. If this is the case, other goals which exist near τ, but on different trajectories will have their probabilities decrease rapidly.

In this case, it can be argued that this is actually the preferred behaviour of the system as mutually-exclusive goals which are not being pursued are eliminated early. However, this is most likely a poor reflection of real world application, where the true goal can be located anywhere in the goal-space. Additionally, the number of candidate goals on the same trajectory as G^* may be minimal once the final observation is processed, making hypotheses more accurate by side effect of domain design.

A minor consideration is that IGRAPH may give a small bonus to facts in the initial hypothesis where $h(G) \approx \tau$ and prior probabilities are equal, as more distant goals are given preference during hypothesis tie breaking (see Algorithm 4, page 80).

In practice, the impact of this behaviour will be determined by the domain designer and problem at hand. For evaluation of IGRAPH, this "optimised" goal-space is only the case in some of the domains tested, such as Driverlog, Rovers and Openstacks, and sometimes only on specific problems. It is unlikely that this will have a noticeable impact on results when it is considered that the accuracy of all heuristics used lowers as the true estimate increases (see Table 4.2). This should negate the effects described above when non-trivial problems are evaluated.

5.8 Plan Sources

In order to perform goal recognition, IGRAPH requires three files: a *domain* file describing the lifted representation of the problem; a *problem* file describing any objects present along with a grounded initial state; and a *solution* file, containing a valid plan for the problem file. Of course, as these are taken from the IPC there is an additional *goal* section in the problem file. This is removed by the test harness used in evaluation, before recognition begins.

The solutions used in evaluating IPC domains come from the results of each associated competition. Each domain has approximately 20 associated problems varying from easiest to hardest (to find a plan in), which will in turn have been solved by a number of planners. For each problem, the plan which has the lowest plan length is selected as the solution for use in IGRAPH. This means that the solutions of multiple planners can be used when performing recognition on a domain. For example, problem 1 may be a solution produced by FF [88], while problem 2 may use a solution from LPG [66]. Naturally, this leads to older planners being used in older competition domains, while more recent competitions represent a closer estimation of the state-of-the-art. Portions of example domain, problem and plan files are shown in Figure 5.1. Descriptions of each domain used in the evaluation of IGRAPH are given in Appendix D. Figure 5.2 shows the number of goals which appear in each domain and problem used in testing.

The use of the FF heuristic in both IGRAPH and the planner which generated the solution used in testing raises the interesting question of *heuristic bias* during recognition. Section 5.18 evaluates the possibility of recognition results being skewed by the observer and agent using the same heuristic for goal-estimation.

5.9 Goal and Action-Space Sizes

To provide context in the following sections, it is useful to describe the size of the goal and action-spaces of the IPC domains tested. Figure 5.3a shows the variance in goal-space size for IPC3 and IPC5 domains, while Figure 5.3b shows the size of the respective action-spaces. Note that these are the respective results

```
(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)))

...
```

```
(define (problem DLOG-2-2-2)
        (:domain driverlog)
        (:objects
                driver1 - driver
                truck1 - truck
                package1 - obj
                s0 - location
                s1 - location
                p1-0 - location
        )
        (:init
                (at driver1 s2)
                (at truck1 s0)
                (empty truck1)
                (at package1 s0)
                (path s1 p1-0)
                (path p1-0 s1)
                (path s0 p1-0)
                (path p1-0 s0)
                (link s0 s1)
                (link s1 s0)
        )
        (:goal (and
                (at driver1 s1)
                (at truck1 s1)
                (at package1 s0))
        )
)
```

```
(walk driver1 s2 p1-2)
(walk driver1 p1-2 s1)
(walk driver1 s1 p1-0)
(walk driver1 p1-0 s0)
(board-truck driver1 truck1 s0)
(drive-truck truck1 s0 s1 driver1)
(disembark-truck driver1 truck1 s1)
```

Figure 5.1: Example domain, problem and solution files in PDDL 2.1, for a small DRIVERLOG problem. This example uses typing of objects, which could be compiled away to an ordinary STRIPS representation.

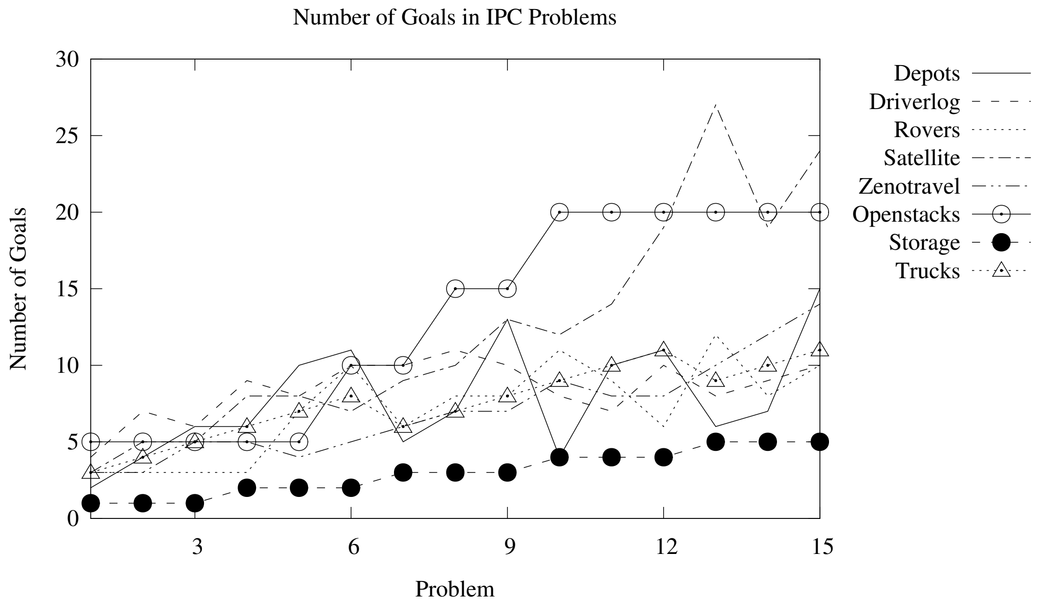


Figure 5.2: The number of goals specified in each respective problem file for each IPC domain used in evaluation. Easier problem numbers require around 5 goals be satisfied, while harder goals require 5–10 goals be achieved. Satellite and Openstacks require even higher numbers — approximately 15–25 goals in each problem.

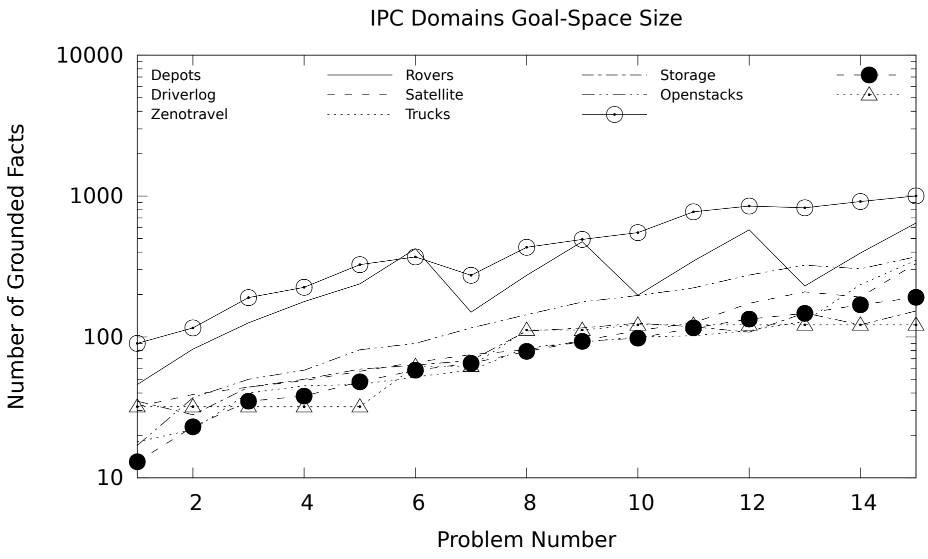
after reachability analysis has been performed. Figures 5.4a and 5.4b show how the reachability analysis reduces the goal and action-spaces on each problem. In some domains this can result in a 90% reduction over a complete model, while in others it makes no difference.

For example, in Depots problem 15, the naïve number of grounded actions is 29232, but after simple reachability analysis becomes 4002. This is caused by a large number of unreachable actions being generated during the grounding process. However, other domains such as Zenotravel offer no such reductions, as all actions generated during grounding are reachable[11].

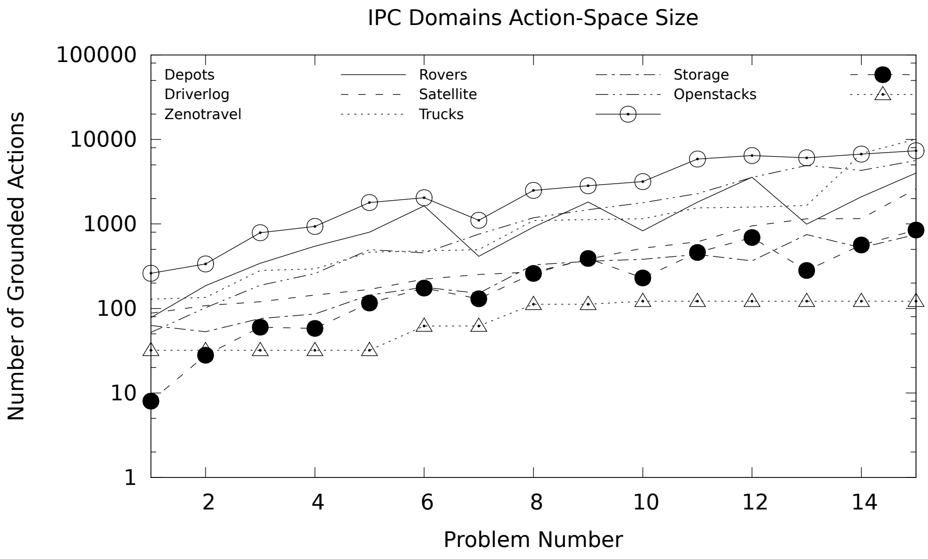
5.10 Intermediate Hypothesis Evaluation

The primary function of any goal recogniser is to determine the observed agent's true goal, $G^* \subseteq S^*$, where S^* is the state in which observation ends. These

[11]This is the case in both Zenotravel and Satellite. In the former, planes can fly directly from an airport to all other airports, making all grounded goals and actions reachable. In the latter, satellites are similarly free to turn and face any target without any ordering constraints.

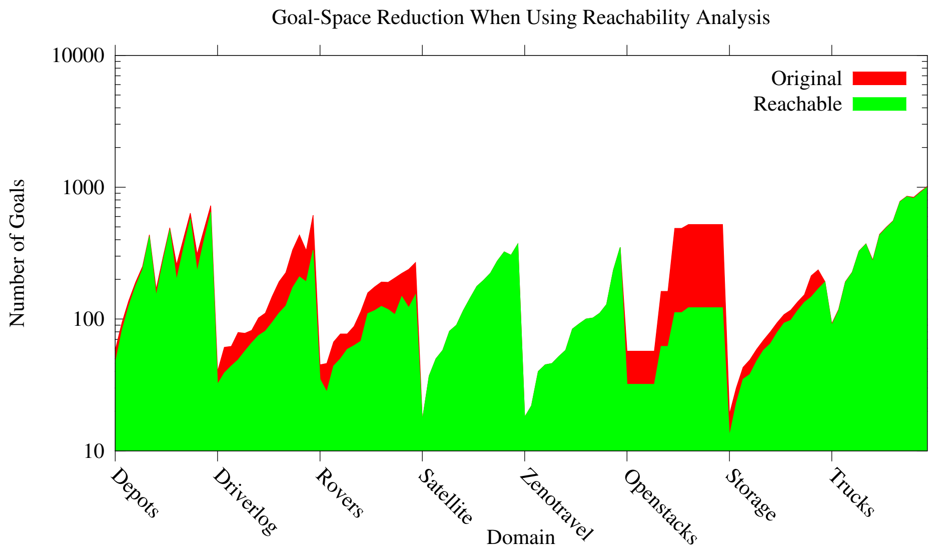


(a) Goal-space sizes

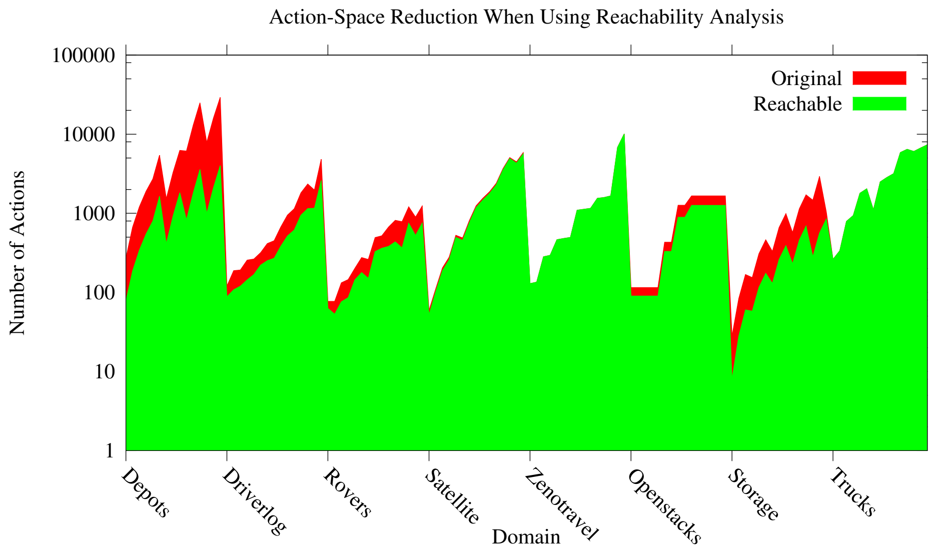


(b) Action-space sizes

Figure 5.3: The size of the goal and action-spaces after reachability analysis in an alternative, overlapping form. Simple problems tend to have $10 - 100$ goals, while harder domains such as DEPOTS and TRUCKS have $100 - 1000$ goals. Action-spaces scale similarly, but often at an order-of-magnitude greater.



(a) Goal-space sizes before and after reachability analysis.



(b) Action-space sizes before and after reachability analysis.

Figure 5.4: The size of the various goal and action-spaces across problems 1-15 for each respective IPC domain, before and after reachability analysis has been performed.

hypotheses are produced after each observation, but are constructed differently based upon whether IGRAPH has been told that the plan has finished.

For clarity, hypotheses which are produced after an observation are referred to as *intermediate* hypotheses, and are produced on the assumption that there may be further observations. These hypotheses are computed using Definition 18 (page 78) and Algorithm 3 (page 79). Hypotheses which are produced in the knowledge that the plan has terminated are referred to as *final* hypotheses. In this situation, the goal must form part of the final state S^*, so hypothesis generation is simply the process of selecting those facts $f \in S^*$ which are most likely to be a goal. These latter hypotheses are covered in Section 5.11.

The intermediate hypotheses produced by IGRAPH are perhaps the most familiar output of the system. Such hypotheses encapsulate the set of goals which the recogniser believes have been most actively pursued by the agent. This is achieved in accordance with the original model presented in Chapter 3, where both the observer and agent are assumed to have optimal heuristics/plans, therefore reducing hypothesis generation to one of tie-breaking those mutually-exclusive goals with equal, maximum probability. This method implicitly extends to the relaxed model presented in Chapter 4, albeit with suboptimal heuristics possible for both agent and observer.

5.10.1 Cross-Domain Results

In order to simplify the visualisation of these results, they are first presented in terms of the *total* F_1 score for each domain and problem. While hiding the progress of recognition accuracy, this serves the purpose of providing a single evaluator which configurations can be compared against. In theory, if configuration A provides better recognition than configuration B on a problem it will result in a higher total F_1 score, as the plan sources are the same.

Figure 5.5 shows these results for all intermediate hypotheses produced during observation, where the total F_1 score for a single problem is simply $\sum_{i=1}^{|P|} \mathrm{F}_1(Hyp_i)$, Hyp_i is the intermediate hypothesis produced after observation i, and $|P|$ is the final length of the observed plan. Results for problems 1–15 are ordered from left to right for each domain. Harder problems tend to have higher scores than lower-numbered problems as there are more observations in each associated plan.

Figure 5.5a displays these results in their raw form, which serves only to show the difference in overall score between domains. Here the results cannot be accurately compared, as the differing plan lengths or accuracy of hypotheses in each domain can appear to weight results to a specific configuration. Therefore, Figure 5.5b provides a more suitable high-level comparison of each configuration on each domain tested. This is achieved by stacking the total F_1 score of each problem tested to reveal the relative accuracy of each configuration.

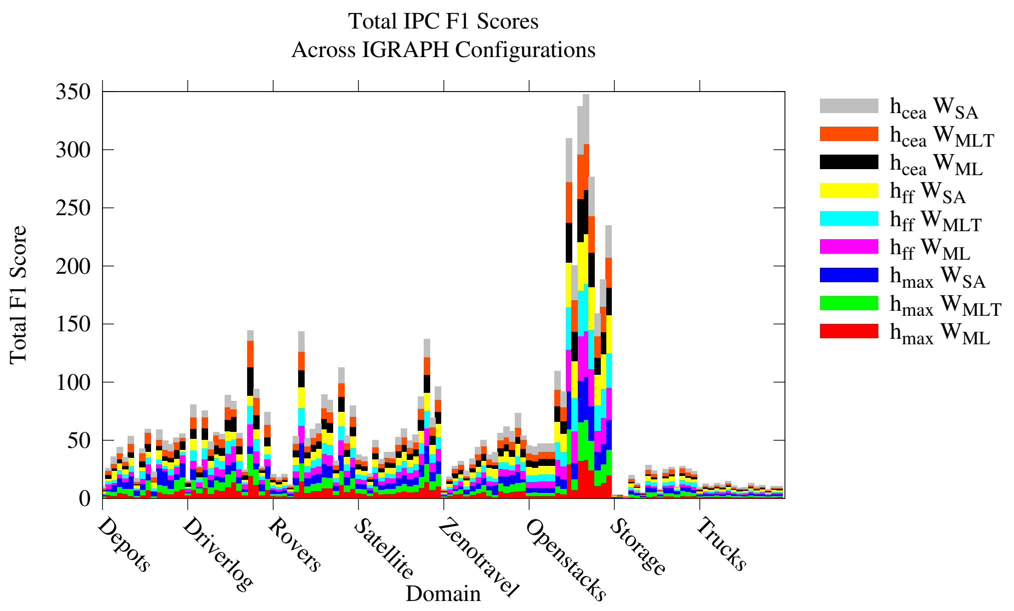
The most obvious feature of this resulting graph is that there is no clear outright "winning" configuration for recognition on generic domains. Some results show a large variation in results, even within the same domain (DRIVERLOG), while others show a near-uniform results across all problems (ROVERS, SATELLITE and TRUCKS).

Domains showing near-uniform results correlate directly with those which contain large numbers of *strictly terminal* facts and goals. This greatly simplifies the recognition process, leading to more uniform hypotheses, regardless of configuration. As this will result in a near-perfect *recall* score, there will be little variation in hypothesis F_1 score (as *precision* will also remain relatively constant). These traits will also be manifested in subsequent results given in this chapter.

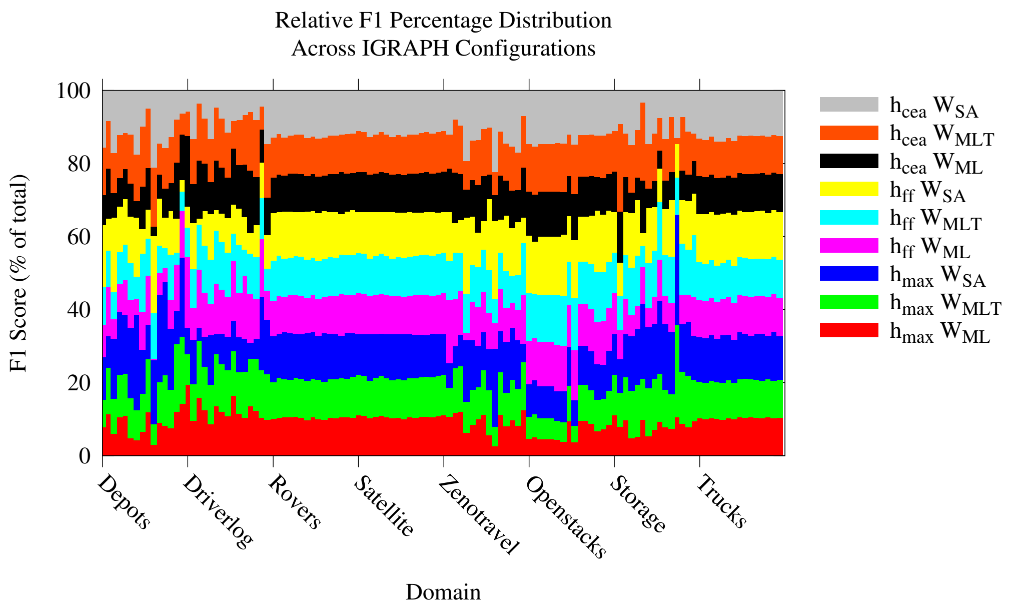
Continuing with a high-level analysis of the IPC results, Table 5.1 shows the overall weighting of the total F_1 score for each configuration relative to each specific heuristic and work function. This shows that the h_{ff} and h_{cea} heuristics outperform h_{max} in all configurations, although the difference between these two former heuristics is negligible. Similarly, the W_{ML} work function is outperformed by W_{MLT} and W_{SA} to a similar level of accuracy. While h_{max} produces a lower overall accuracy, the score is still sufficiently high to merit further investigation. Section 5.19 enumerates possible reasons for this performance.

The raw results for each configuration are given in Table 5.2. These show that h_{ff} and h_{cea} each outperform other heuristics in three of the eight domains tested. The pairing of $\langle h_{max}, W_{SA} \rangle$ is the only configuration where h_{max} provides better results on a specific domain (DEPOTS and STORAGE). The pairing of $\langle h_{ff}, W_{SA} \rangle$ narrowly beats others in being the best overall performing configuration, although this is largely caused by a high score in OPENSTACKS.

The similarity between configurations using h_{ff} and h_{cea} is caused by most domains having approximately equal results across each work function configuration, indicating that in practice both of these heuristics have equal performance (although not necessarily equal estimates). However, the features of each heuristic are visible in some specific domains. For example, h_{cea} outperforms h_{ff} on DEPOTS where the delete-relaxation causes poor estimates on the crate-stacking subproblem.



(a) The total, stacked F_1 score for all configurations of IGRAPH across all IPC domains.



(b) Stacked intermediate hypothesis results, using total F_1 score as a percentage of the total achieved score.

Figure 5.5: F_1 scores for all *intermediate hypotheses* produced across all domains and configurations. Figure 5.5a demonstrates the difference in (non-normalised) F_1 scores across each domain. Figure 5.5b shows these results as a percentage of the total F_1 score for each problem.

Configuration	h_{max}	h_{ff}	h_{cea}	Total
W_{ML}	9.33	10.92	10.86	31.10
W_{MLT}	10.34	11.27	11.64	33.25
W_{SA}	11.39	12.31	11.94	35.65
Total	31.05	34.50	34.44	100.00

Table 5.1: The percentage of total F_1 score which each configuration of IGRAPH achieved, in terms of the sum of these scores.

Domain		h_{max}			h_{ff}			h_{cea}		**Total**
	W_{ML}	W_{MLT}	W_{SA}	W_{ML}	W_{MLT}	W_{SA}	W_{ML}	W_{MLT}	W_{SA}	
Depots	55.53	76.72	**103.68**	53.37	53.47	67.29	57.68	72.88	71.06	611.68
Driverlog	116.61	119.43	83.27	116.05	104.68	80.85	**125.09**	123.8	83.62	953.4
Rovers	92.29	95.86	110.94	93.13	95.53	110.45	92.78	94.93	**111.07**	896.98
Satellite	91.71	93.5	102.31	92.86	93.47	**102.52**	92.96	93.49	102.28	865.1
Zenotravel	57.21	64.82	73.85	61.85	64.03	78.27	65.87	70.7	**82.18**	618.78
Openstacks	189.45	208.26	245.4	294.85	311.74	**349.86**	274.29	294.09	318.34	2486.28
Storage	18.52	31.97	**39.23**	18.09	31.52	31.64	17.4	29.89	27.41	245.67
Trucks	17.13	17.16	20.76	17.13	17.12	**22.05**	17.12	17.12	21.63	167.22
Total	638.45	707.72	779.44	747.33	771.56	**842.93**	743.19	796.9	817.59	6845.11

Table 5.2: The total F_1 score for each domain and configuration across all 15 test problems. The best performing configurations are highlighted in bold.

While providing an overall impression of each configuration, these results only provide information on how IGRAPH performed after observation has completed. The following section breaks down these results into the performance of each configuration as observation progresses.

5.10.2 Accuracy During Observation

With an overview of the intermediate results given, this section will examine how the accuracy of hypotheses changes *during* plan observation. The expected outcome of the heuristic-based observation process is that higher posterior probabilities will be assigned to those goals which have had their heuristic estimate lowered by the greatest amount since observation began. This equates to an increase in accuracy as more observations are processed.

As plan-length varies massively from problem-to-problem, all results for a given problem have been normalised against the length of the respective plan. This enables a comparison of hypotheses for differing problems, constructed after the same percentage of total observations. However, unlike the previous results overview, this section enumerates the *precision*, *recall* and F_1 scores.

Figures 5.6 and 5.7 show the average F_1 score for all hypotheses at problem instantiation, 25%, 50%, 75% and 100% of plan completion, on each IPC domain tested. Note that at all stages (including 100% of the plan being observed), IGRAPH is never told that the plan is over or alternatively that there will be more observations. For *intermediate* hypothesis generation, this latter case is always assumed to be true. However, the standard *initial hypothesis* is used for values at 0% of observation, as a means of showing how observations affect the initial goal hypothesis. This will be covered explicitly in Section 5.15.

The F_1 scores show that DEPOTS, DRIVERLOG, OPENSTACKS and ZENOTRAVEL display the expected *heuristic convergence* behaviour, whereby more observations lead to an increase in hypothesis accuracy across all configurations. STORAGE shows some improvement in accuracy if the appropriate configuration is used. The remaining domains show no improvement in the F_1 score for any configuration. This is naturally a disconcerting result, but can be explained when the individual P+R graphs are analysed separately. Figures 5.8–5.9 and 5.10–5.11 respectively show these results.

In the case of precision, half of the domains tested exhibit some form of convergence upon the correct number of goal literals (DEPOTS, DRIVERLOG, ZENOTRAVEL and OPENSTACKS). Recall that precision should increase if an entire sub-goal-space is omitted from a hypothesis due to the *all false* goal becoming the most likely candidate, although it is possible that the greedy hypothesis construction algorithm eliminates other goal candidates through mutex relations. While these results are far from perfect (precision scores never exceed 50% accuracy), they do indicate that IGRAPH is performing as expected. This behaviour can be construed as recognising *negative evidence* — where a lack of evidence is used to support a hypothesis — which is one of the key features of both the heuristic-based model presented and of Goldman *et al's* widely accepted model [60, 70].

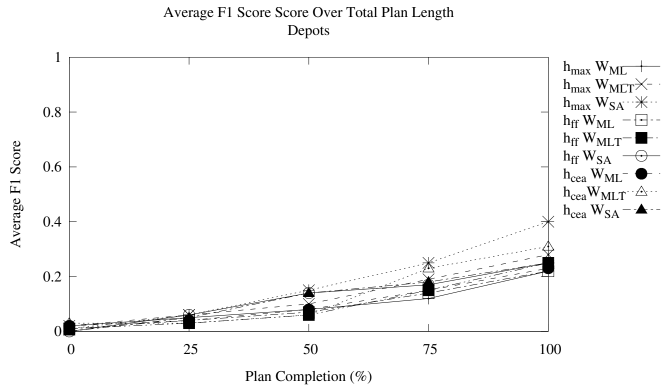
Delving into individual results, the ROVERS, SATELLITE and TRUCKS domains all show near-perfect *recall* results across all configurations, due to the often almost exclusive presence of *strictly terminal* facts in the agent's goal. As stated in Section 4.4.1, once achieved these goals are always added to all future hypotheses, which maintains the high recall score. Unfortunately, this is at the cost of precision as hypotheses will often contain surplus, but ultimately indistinguishable, ST goals. This behaviour is the source of the near-uniform results shown in Figure 5.5 for these domains. The corresponding F_1 score for TRUCKS is extremely low due to the poor precision scores, caused by a large number of sub-goal-spaces in $\mathcal{G}^V$, all of which contribute to the hypothesis size.

The remaining domains (Driverlog, Depots, Zenotravel, Storage and Openstacks), all demonstrate a varying degree of convergence upon the true goal over the course of observation. However, the work function applied appears to cause differing results based on the target domain. For instance, Driverlog under-performs when using the W_{SA} function, regardless of the heuristic used, while in Storage this function performs best when paired with h_{max} rather than a more accurate heuristic.

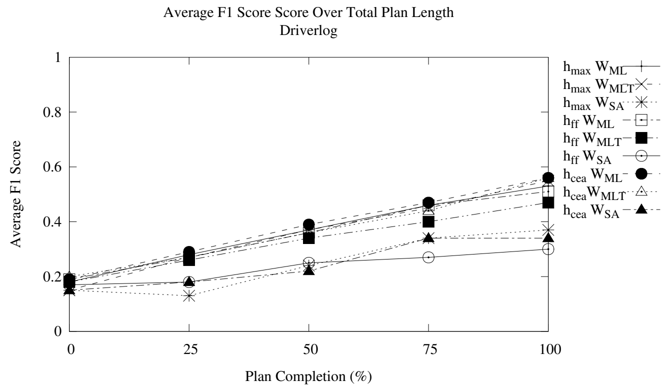
Openstacks shows an interesting recall-rate. Despite containing ST facts[12], the goal is not converged upon until the last 25% of observation, with configurations using h_{ff} and h_{cea} performing equally-well. This indicates that the heuristics used in testing may have trouble detecting movement toward this goal, or that it is obscured by movement toward a mutex goal. This is also the only domain to demonstrate the expected behaviour regarding h_{max} being outperformed by all other configurations.

A further interesting result is that of Rovers. Here, there is a dip in recall accuracy at 25% of observation. This is caused by the initial hypothesis being correct, but the heuristic estimates to the relevant goals being less accurate in the initial stages of observation, causing these literals to be dropped from hypotheses. These particular literals are largely related to the position of rovers, which can appear in the true goal, but are often the same as the start position, meaning estimates increase in the early stages of the plan. The remaining 75% of observation allows these estimates to again be converged upon. Similar behaviour can be seen in Trucks, albeit on a smaller scale.

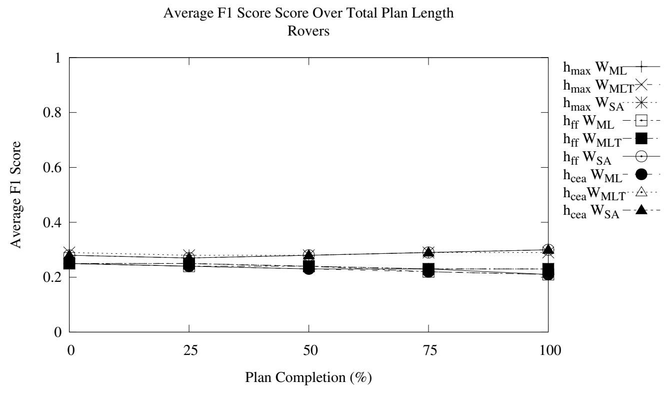
[12]Once an order is complete in Openstacks, it is considered `shipped`, which is usually the goal of the problem.



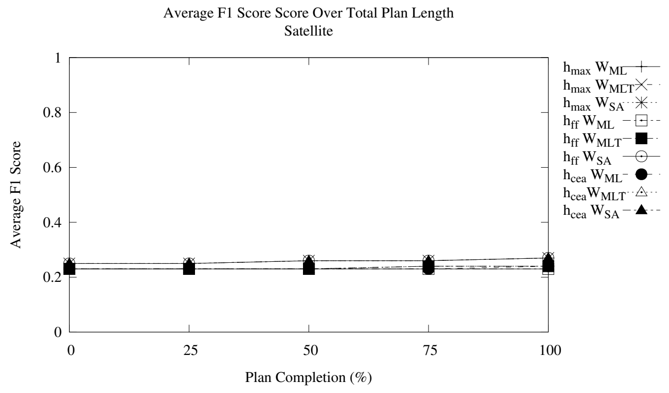
(a) Depots



(b) Driverlog

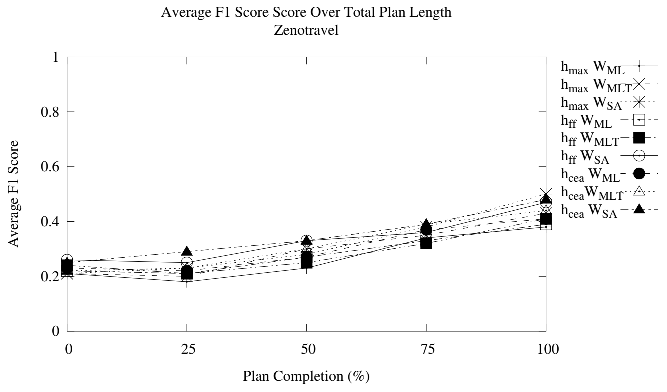


(c) Rovers

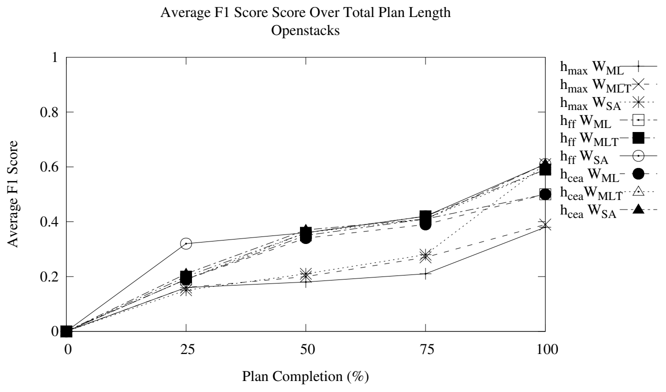


(d) Satellite

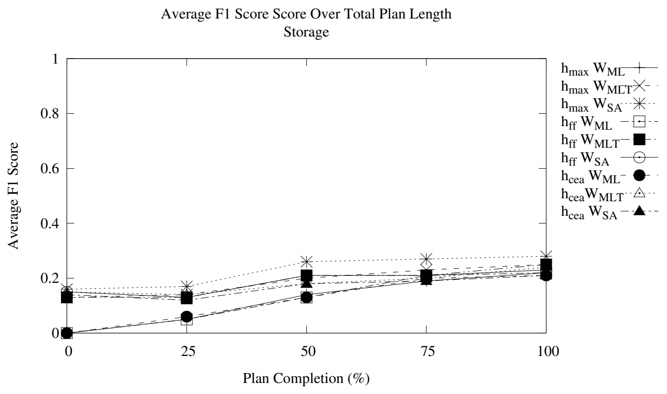
Figure 5.6: The average F_1 score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



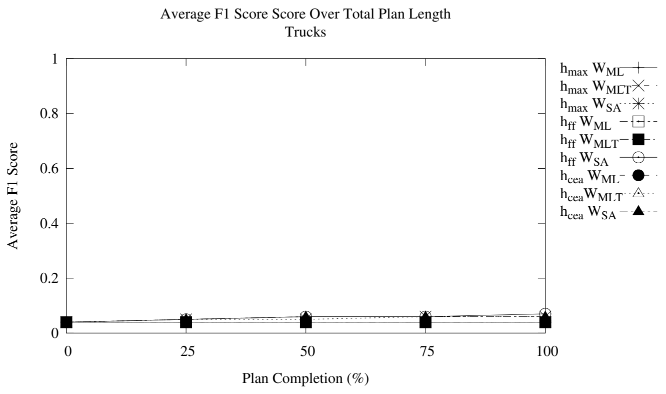
(a) Zenotravel



(b) Openstacks

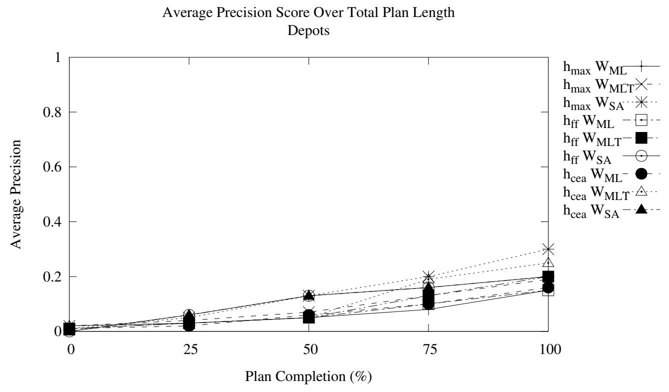


(c) Storage

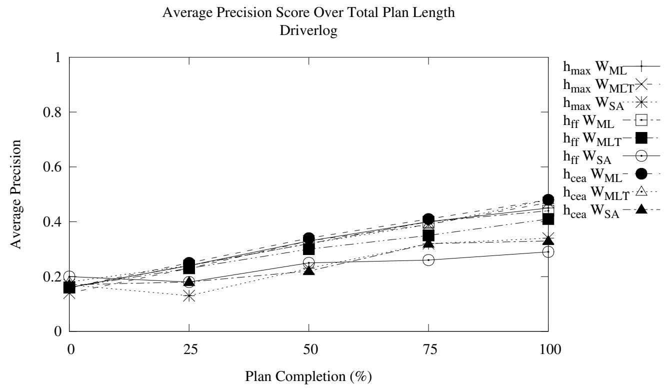


(d) Trucks

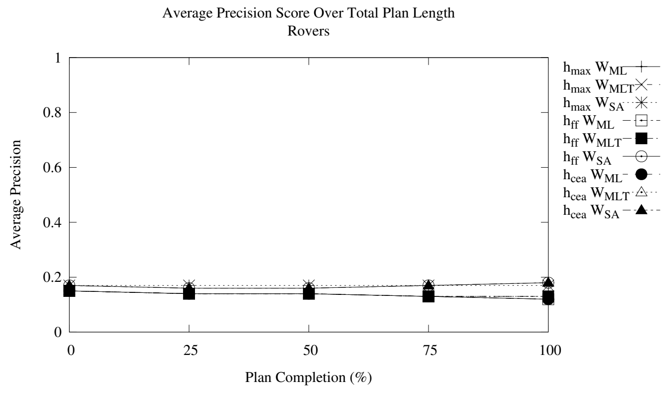
Figure 5.7: The average F_1 score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



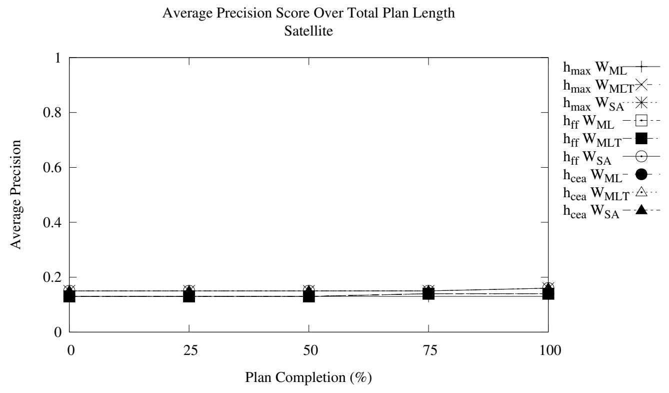
(a) Depots



(b) Driverlog

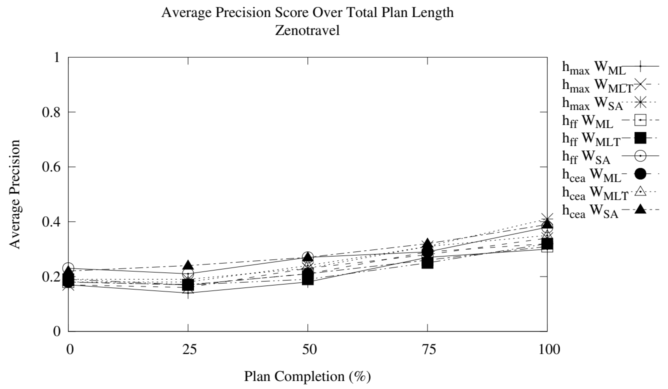


(c) Rovers

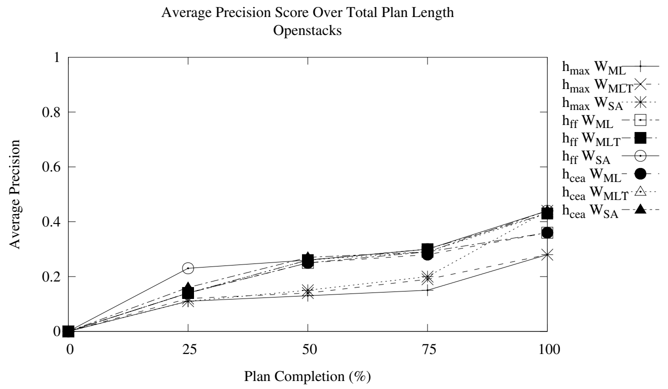


(d) Satellite

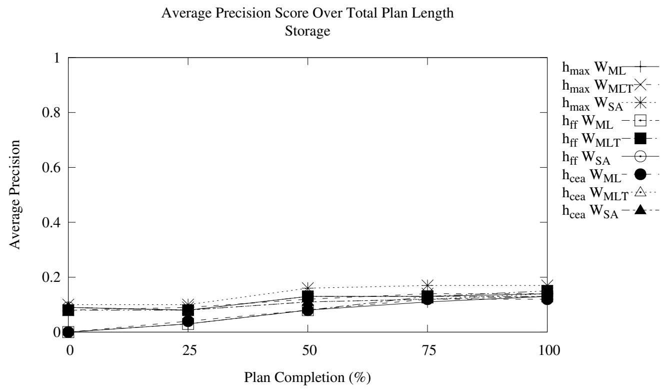
Figure 5.8: The average *precision* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



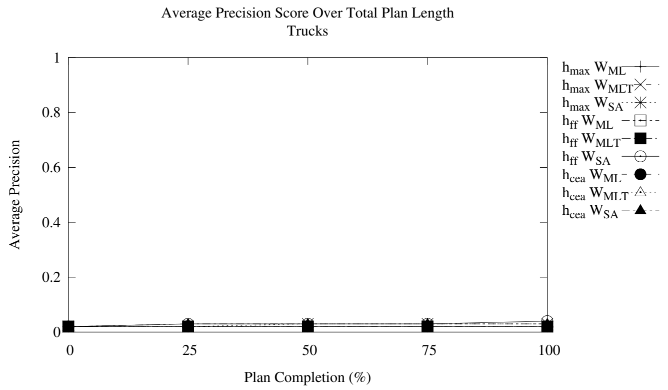
(a) Zenotravel



(b) Openstacks

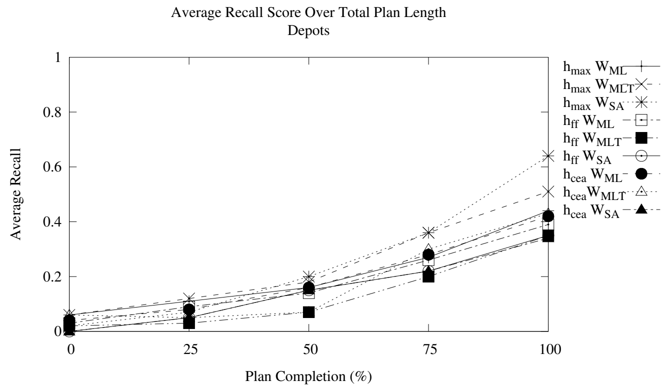


(c) Storage

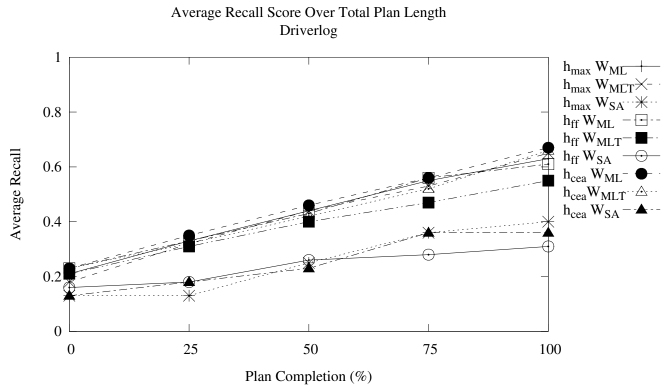


(d) Trucks

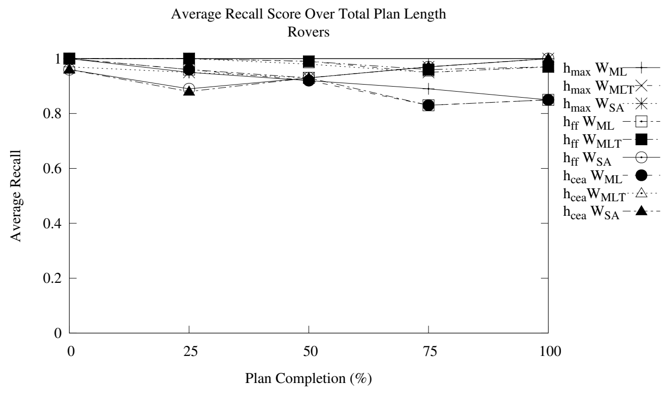
Figure 5.9: The average *precision* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



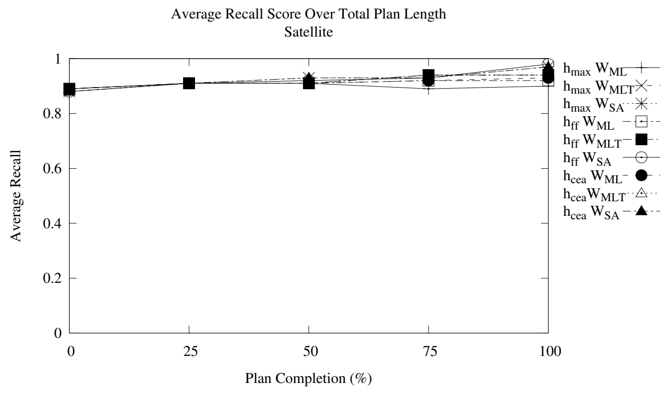
(a) Depots



(b) Driverlog

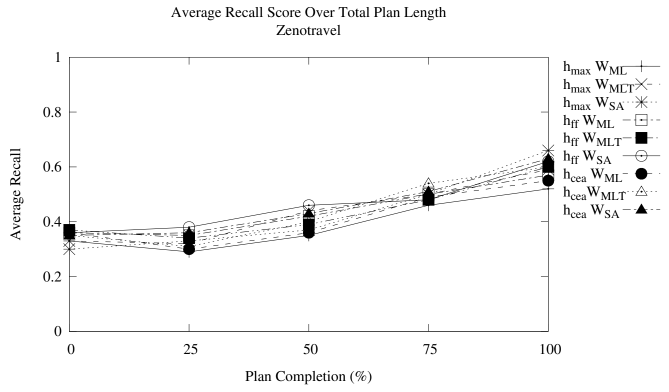


(c) Rovers

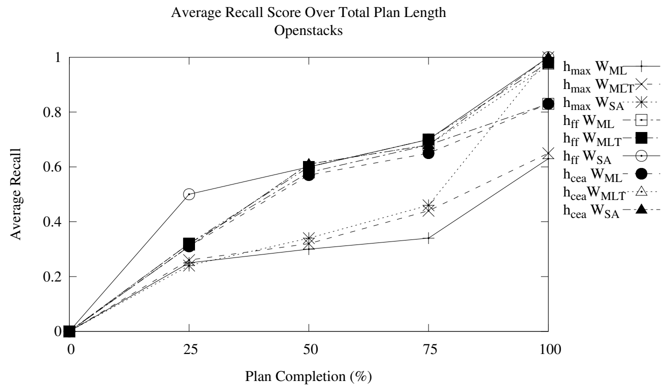


(d) Satellite

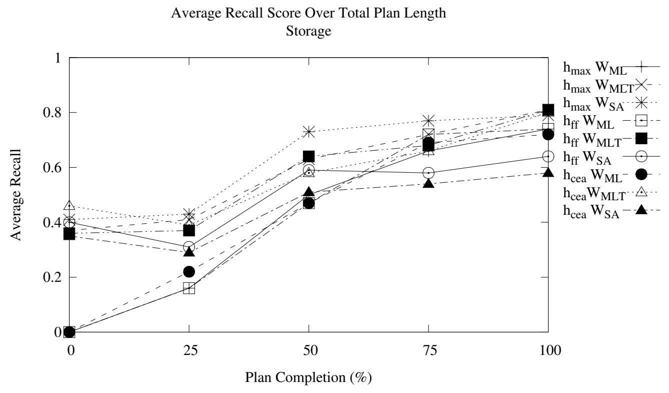
Figure 5.10: The average *recall* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



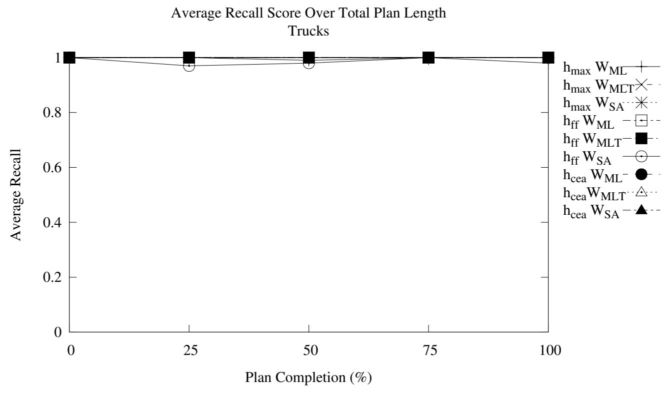
(a) Zenotravel



(b) Openstacks



(c) Storage



(d) Trucks

Figure 5.11: The average *recall* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.

These results show that heuristic convergence occurs for most domains and configurations of IGRAPH, with the exception of those containing ST facts. However, these do not provide any information on the *rate* at which recognition occurs. As the complete GR model should theoretically converge on the goal at a linear rate (due to an optimal heuristic and subject), this convergence rate can be a useful indicator of how well IGRAPH is performing.

5.10.3 Heuristic Convergence Rate

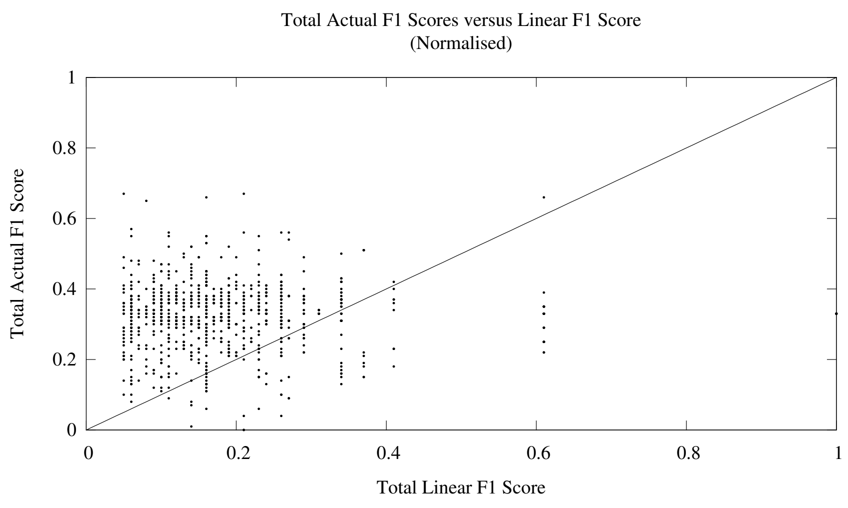
The P+R/F_1 results shown in Figures 5.6 – 5.11 indicate that in general, scores will increase *linearly* with the number of observations processed. That is, if 50% of the plan has been observed, the resulting F_1 score will be around 50% of the final score. This section will explore how accurate this statement is.

The linear F_1 score for a problem can be computed as the sum of achieving a score of $F_1 = \frac{i}{|P|}$ after each observation, where $|P|$ is the length of the actual plan and i is the index of the current observation. Equation 5.5 shows the full form of this.

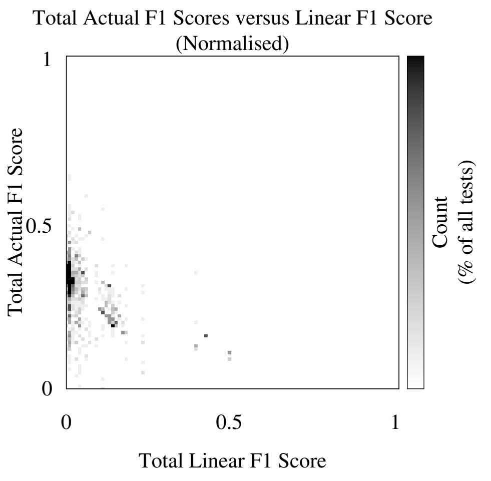
$$\mathrm{F}_{1_{linear}} = \sum_{i=1}^{|P|} \frac{i}{|P|} \tag{5.5}$$

Figure 5.12a shows the normalised total F_1 score for the actual hypotheses produced by IGRAPH compared against the equivalent F_1 score for each problem. The results presented cover all configurations of IGRAPH and have been rounded to the nearest 2d.p. All points which lie above the X/Y axis indicate that the total F_1 score achieved was higher than the equivalent linear score, while those below this line indicate the opposite. Figure 5.12b shows these same results in the form of a heatmap, in order that duplicate results are clearer.

From these two figures it can be seen that IGRAPH is producing results which exceed the potential linear score. While this cannot be used as an indicator of predictive power or accuracy, it does confirm that the accuracy of hypotheses produced is generally better than the total linear score. The percentage of tests with a higher overall F_1 score is given in Table 5.3. These echo the results shown in Figure 5.12, in that a high percentage of all observation-traces exceed the linear rate. However, with the exception of W_{ML}, which has been shown to be the poorest performing work function, there is little difference in the performance of W_{MLT} and W_{SA} regardless of the heuristic used.



(a) A point-cloud of all results produced by IGRAPH during all IPC tests. Results above the line indicate that the actual score from observation exceeded the total linear score.



(b) A heatmap of all IPC results, where the brightness of a cell indicates the number of tests which share the same X/Y values.

Figure 5.12: The total F_1 scores for all IPC domains and configurations, when normalised against the total potential F_1 score for each problem. Figure 5.12a shows these results as a simple point-cloud (with results rounded to the nearest 2 decimal-places). Figure 5.12b shows this same data as a heatmap, where brighter colours indicate the number of results which shared the same value. In both cases the x-axis is the total *linear* F_1 score for a problem, while the y-axis indicates the true F_1 score of the problem.

Configuration	h_{max}	h_{ff}	h_{cea}
W_{ML}	80.00	85.00	85.00
W_{MLT}	89.17	90.00	90.83
W_{SA}	91.67	88.33	89.17

Table 5.3: The percentage of tests across all configurations of IGRAPH which achieved a total F_1 score greater than that of the *linear* total F_1 for the respective problem.

5.10.4 Hypotheses as Final State Predictions

The previous section has analysed how well IGRAPH performs in predicting the final goal of the agent during observation. In these *intermediate* hypotheses, the score for precision was consistently lower than that of recall, which in turn brought down the F_1 score for each respective problem.

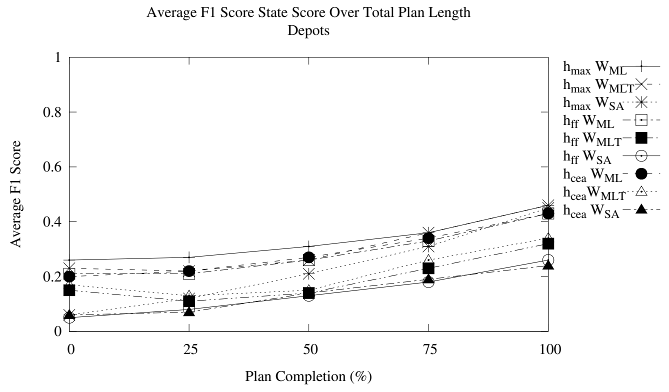
The cause of these low precision scores is that given a set of goals, IGRAPH does not distinguish between those which are more likely to be the end goal rather than simply a standard goal. While the domain analysis used in constructing the initial probability distribution does make some assumptions about what behaviour a "good" goal candidate will show, the hypothesis extraction algorithm bases its output purely on evidence observed.

This algorithm (page 79), extracts a single maximum-probability member from each sub-goal-space in the multivariate goal-space for inclusion in the hypothesis. Therefore, as each sub-goal-space encapsulates a set of mutually-exclusive facts detected at runtime, a hypothesis has the potential to be a complete *relaxed state* prediction. This section briefly analyses how the previously presented results change when hypotheses are viewed in this context, such that $G^* = S^*$. S^* itself is computed simply by applying all observations in the plan, in order, from the initial state.

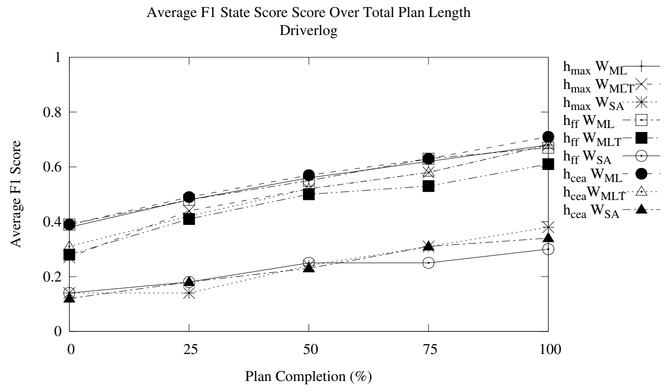
Figures 5.13–5.18 show the average F_1, precision and recall scores, when the same *intermediate* hypotheses generated previously are considered as states rather than goals. As is to be expected, the *intermediate-state* scores are higher across all domains, although only marginally in some cases. Table 5.4 shows the average increase in total F_1 score, across all configurations, over the original intermediate goal hypotheses.

Problem	Depots	Driverlog	Rovers	Satellite	Zenotravel	Openstacks	Storage	Trucks
1	162.7	34.7	86.1	39.6	16.3	230.0	179.8	117.5
2	24.6	5.8	118.8	28.3	20.5	228.2	134.2	101.5
3	41.4	82.9	126.4	65.4	20.9	224.4	210.8	109.6
4	110.5	1.4	163.9	35.5	80.8	224.4	97.5	101.2
5	150.5	38.6	89.3	45.8	211.4	224.4	160.1	102.8
6	309.3	0.4	58.9	59.0	32.9	193.4	615.5	106.5
7	465.7	18.1	100.4	71.1	52.0	249.4	133.2	101.0
8	120.1	26.5	102.9	59.2	31.0	117.3	104.9	116.8
9	169.1	30.8	113.9	55.8	98.9	279.1	363.5	109.7
10	1118.9	36.9	93.6	68.4	116.0	86.0	198.4	115.2
11	153.4	244.7	109.6	42.4	26.1	82.6	162.7	110.5
12	361.8	10.9	135.9	33.6	45.5	116.0	-	104.0
13	52.7	42.8	74.6	25.6	73.9	243.8	191.7	106.6
14	62.0	171.6	118.3	43.0	63.9	197.1	201.6	121.4
15	131.6	81.3	85.1	41.3	144.8	143.2	225.0	102.2

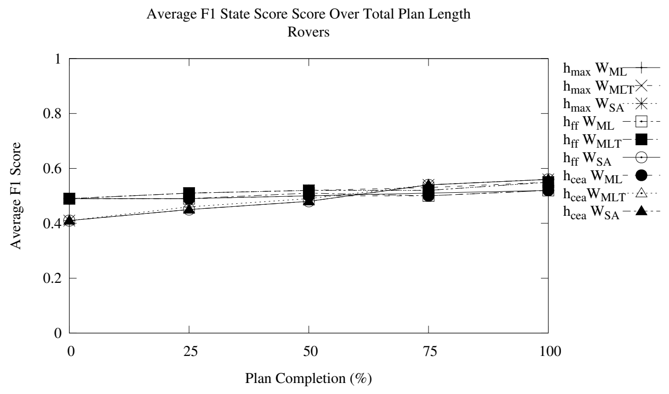
Table 5.4: The average percentage increase in the overall F_1 score, across all configurations, when hypotheses are regarded as final state predictions rather than final goal predictions. Entries with a '-' indicate results where the total F_1 score for goal hypotheses is zero.



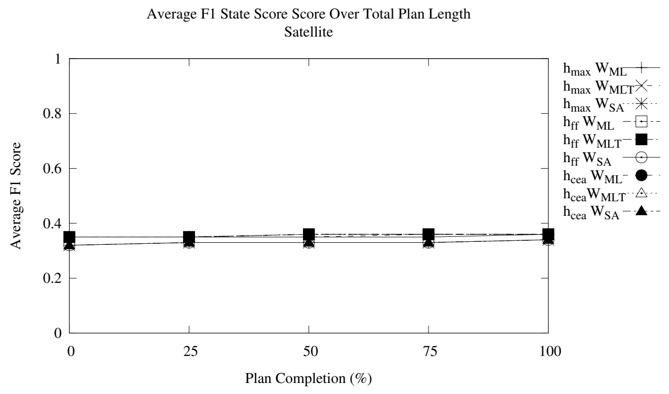
(a) Depots



(b) Driverlog

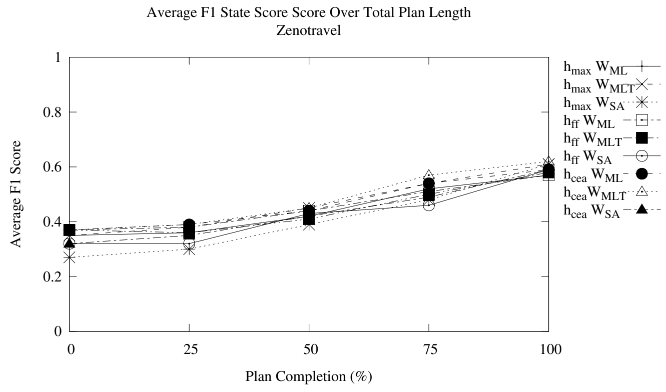


(c) Rovers

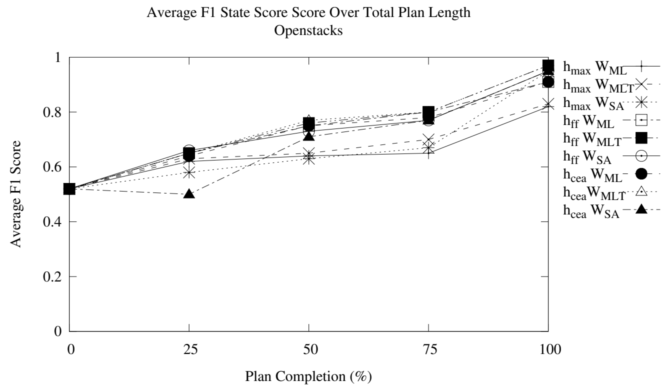


(d) Satellite

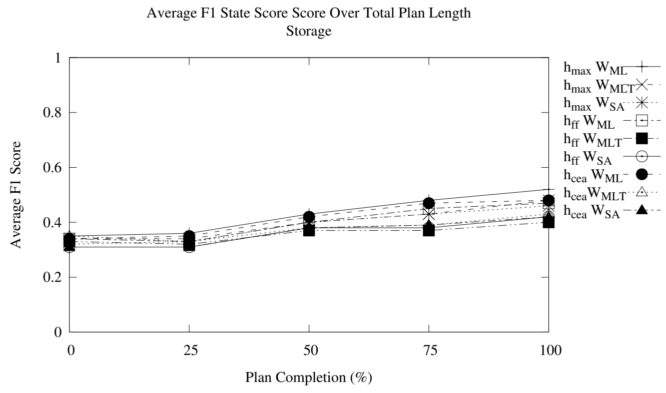
Figure 5.13: The average *state* F_1 score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



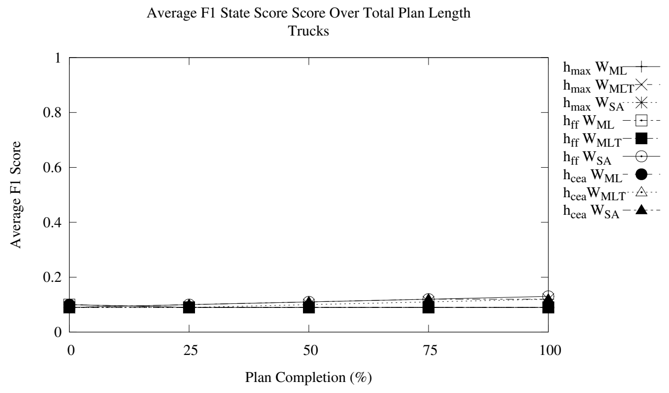
(a) Zenotravel



(b) Openstacks

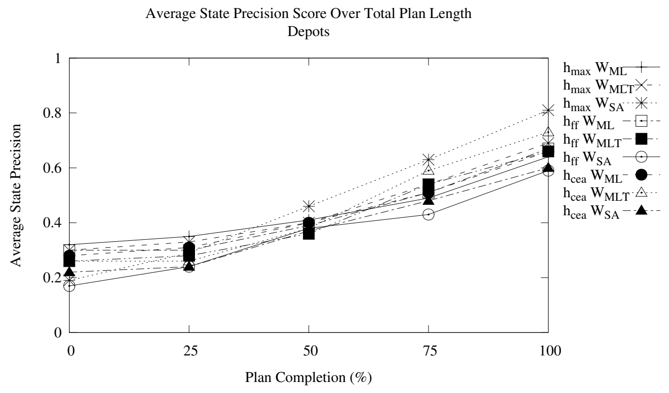


(c) Storage

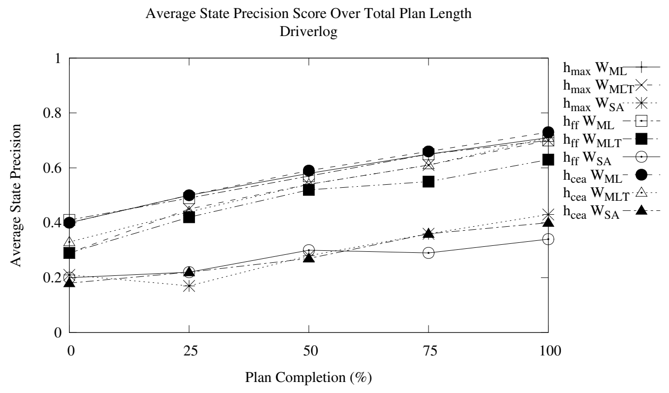


(d) Trucks

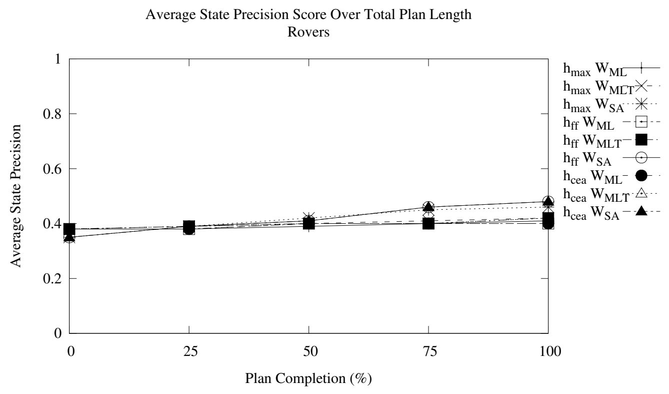
Figure 5.14: The average *state* F_1 score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



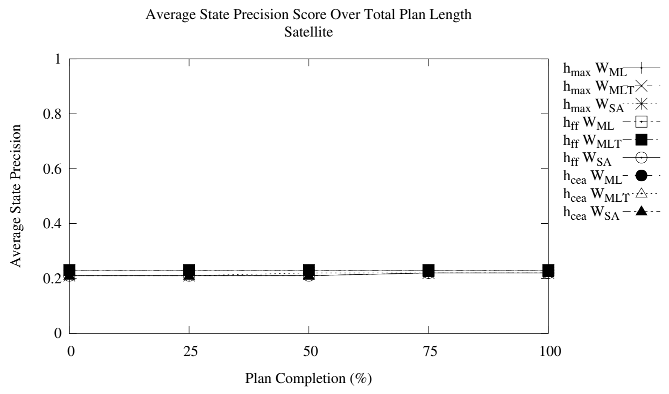
(a) Depots



(b) Driverlog

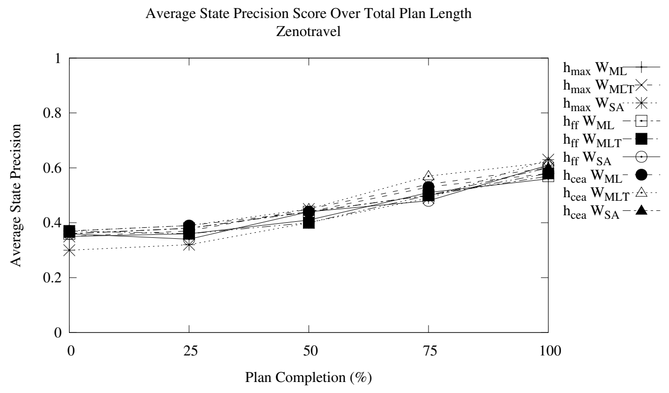


(c) Rovers

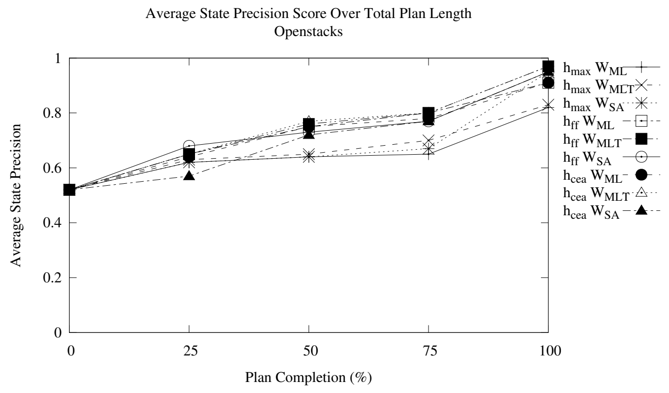


(d) Satellite

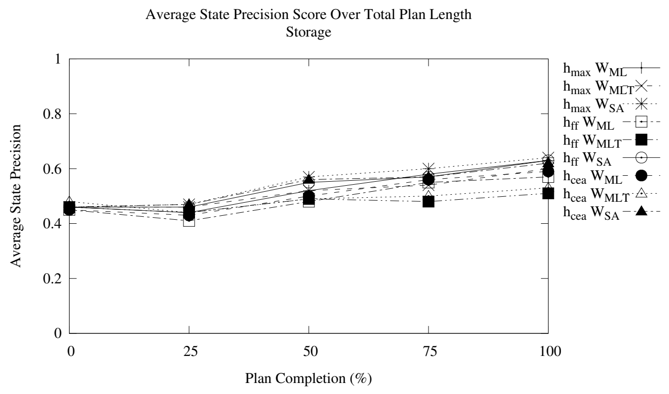
Figure 5.15: The average *state precision* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



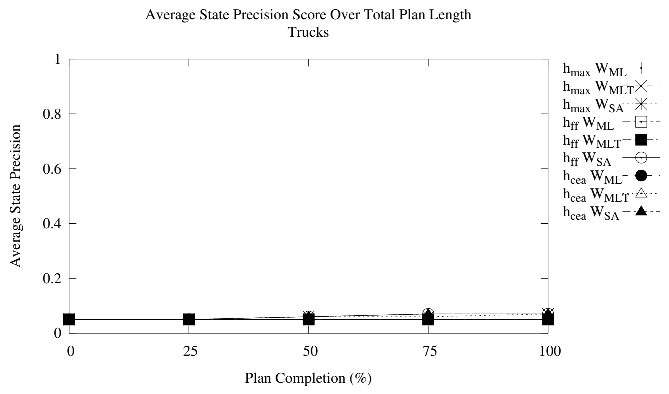
(a) Zenotravel



(b) Openstacks

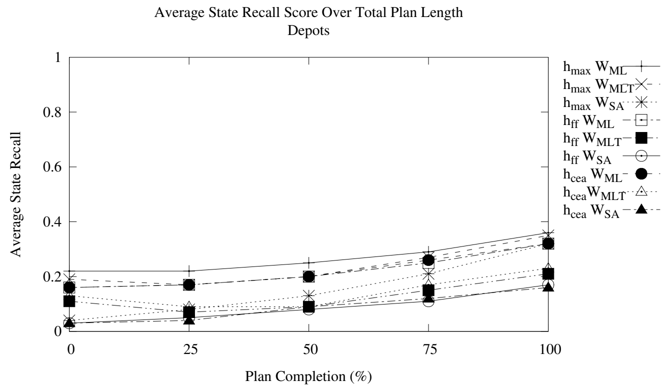


(c) Storage

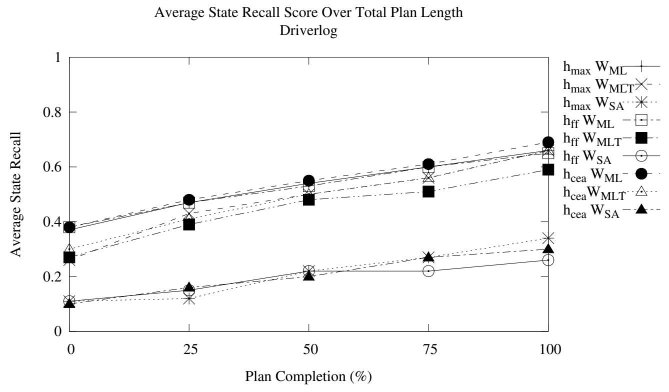


(d) Trucks

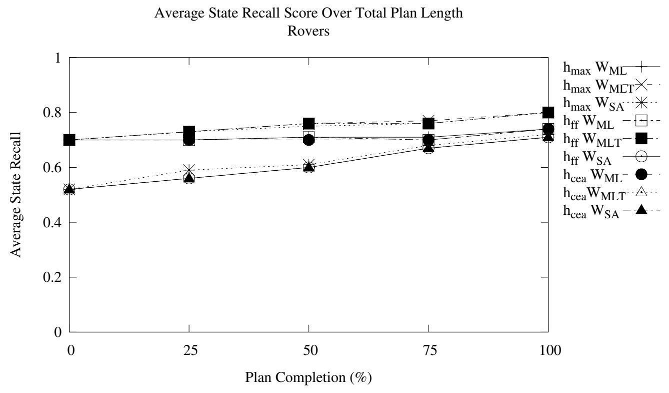
Figure 5.16: The average *state precision* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



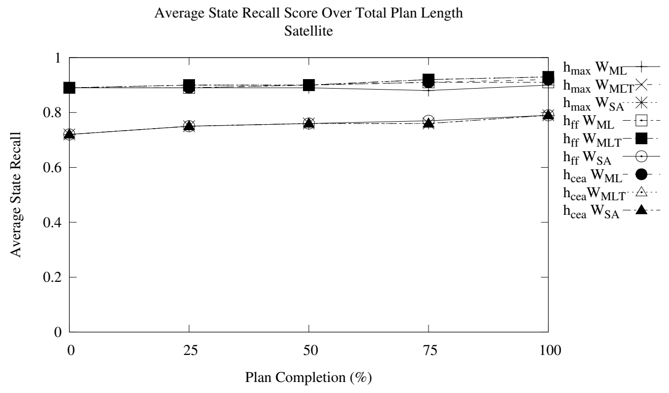
(a) Depots



(b) Driverlog

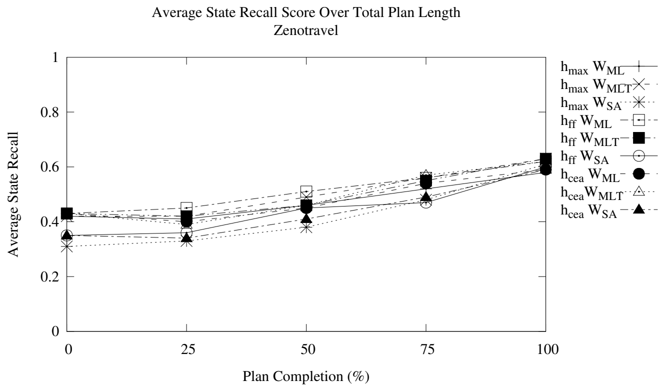


(c) Rovers

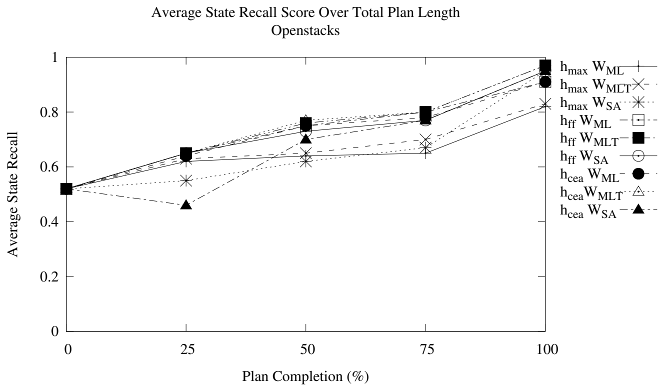


(d) Satellite

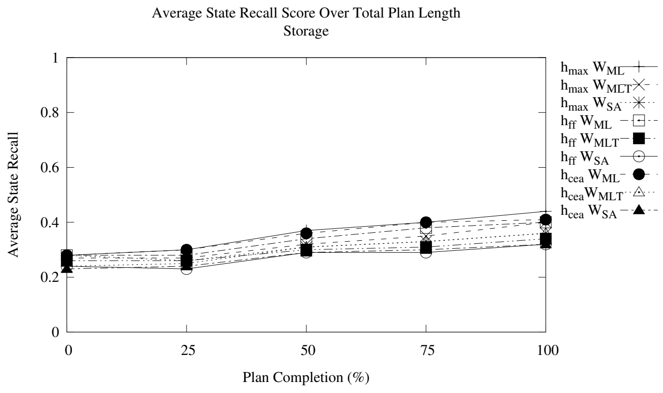
Figure 5.17: The average *state recall* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.



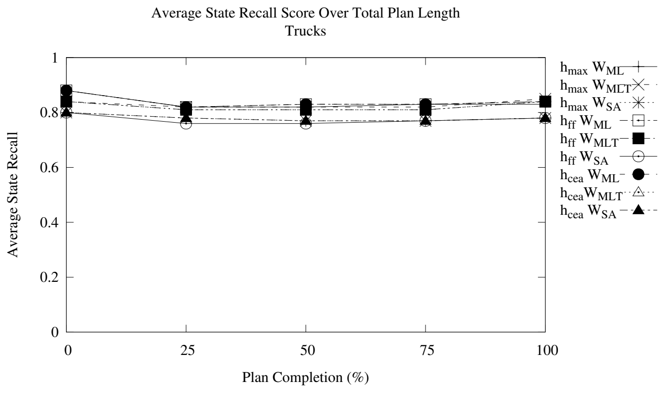
(a) Zenotravel



(b) Openstacks



(c) Storage



(d) Trucks

Figure 5.18: The average *state recall* score for each domain and configuration at 25%, 50%, 75% and 100% of plan completion.

While these results may appear impressive, they must be regarded with some scepticism. In some domains it is possible to receive a high P+R score simply by outputting the initial state as the hypothesis, due to minimal state changes between observations. For example, if the initial state contains 10 facts of which 8 will be ignored by the agent, an *intermediate state* hypothesis can achieve a score of 0.8 by saying that the initial state is the goal.

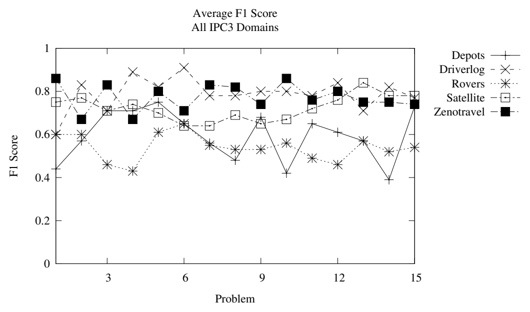
Therefore, while clearly not the preferred output of a goal recogniser, in some applications such as those where a human operator is involved, inferring side-effects or partial states may be of additional use in determining the agent's goal. Future work could potentially explore how the correct inference of one non-goal literal can assist in predicting true-goal literals.

5.11 Final Hypothesis Evaluation

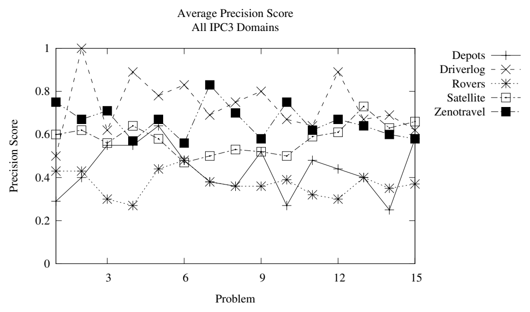
As explained in Section 4.3.2, *final hypotheses* are constructed from those facts true in the final observed state, S^* (the observer is assumed to have been told that observation is complete). Therefore, the probability associated with each goal is no longer necessary, reducing final hypothesis extraction to one of eliminating goals present in S^*.

As described in Section 4.3.2, IGRAPH uses the *stability* of a goal $\Upsilon(G)$ to determine those facts in S^* which are suitable for the final hypothesis $Hyp_\Omega \subseteq S^*$. Any fact which has a stability of at least $0 < \Upsilon(G) \leq \eta \leq 1$ is added to Hyp_Ω.

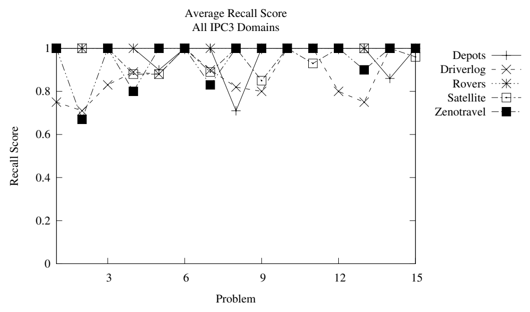
Given a value of $\eta = 1$, final hypotheses will contain only those facts which have been added to the state and never deleted. This value is in keeping with the assumption that the agent will always endeavour to keep a goal literal true once achieved. Figure 5.19 shows the precision of Hyp_Ω under these conditions across all IPC domains. Note that the configuration of IGRAPH is irrelevant, as the plan used in observation is always the same, leading to the same final-state and thus the same final hypothesis. That is, given that the goal state is known, there is no need to run the greedy hypothesis extraction algorithm which could lead to different hypotheses through tie-breaking.



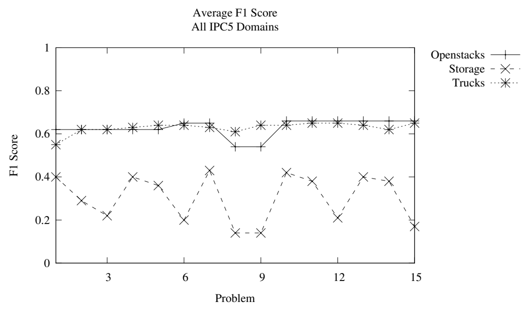
(a) IPC3 domains F_1 score



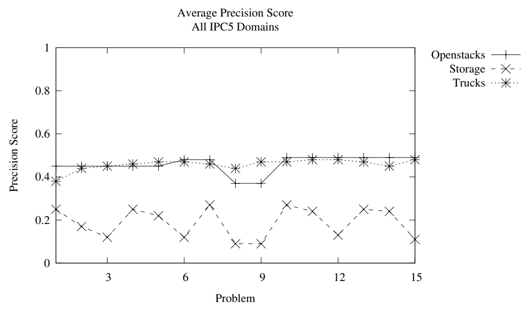
(b) IPC3 domains *precision* score



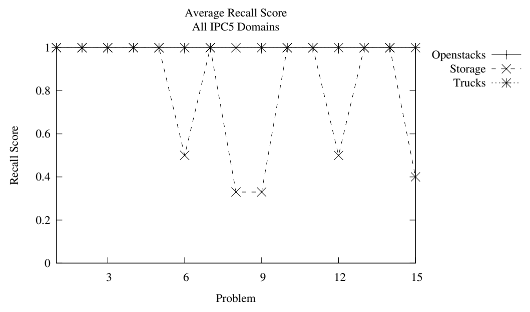
(c) IPC3 domains *recall* score



(d) IPC5 domains F_1 score



(e) IPC5 domains *precision* score



(f) IPC5 domains *recall* score

Figure 5.19: The F_1, precision and recall scores for final hypotheses, produced at the end of observation. Results which are not visible have perfect scores (e.g. the recall score for OPENSTACKS and STORAGE is always 1).

As the results for $\eta = 1$ show, the average F_1 score varies from approximately 0.2 to 0.85 across the various domains (see Table 5.5 for averaged F_1 results). As always, lower F_1 scores are caused by the hypothesis either containing too many goals (too many non-goal facts with $\Upsilon(G) \geq \eta$), or by not proposing all true-goal literals (at least one true goal has $\Upsilon(G) < \eta$).

For the majority of domains tested this trade-off is clear — domains which exhibit goals where $\Upsilon(G) \geq \eta$ often contain many side-effects which will also have high values for *stability*. For instance, in OPENSTACKS the goal is to have each order become `shipped`, however, prior to this the order must also be `made`. Therefore, while a `shipped` goal will always be added to Hyp_Ω, a corresponding `made` fact will also be included, lowering the precision and F_1 scores.

For other domains such as DRIVERLOG, there is the potential for the true goal to include cyclic goals in which the agent must return a resource such as a truck to its original location, or to another location which has previously been visited during the plan. As such, the *recall* score is lower than in all domains other than STORAGE. However, by virtue of these trucks/drivers being part of the goal, the *precision* score is high. STORAGE itself exhibits this same behaviour, but has lower P+R scores as not *all* sub-goal-spaces will contain a goal literal (leading to a low precision score).

Placing the assumption of $\eta = 1$ on the final goal is in line with both the agent-related assumptions presented in Chapter 3 and many previous classical GR models, wherein the achievement of the (often single literal) goal indicates the immediate termination of the plan [18, 38, 70, 146]. However, in this case goals can be achieved throughout a plan, rather than only on the final observation.

Given that this is the expected behaviour, all results for *recall* should be equal to 1. Yet, as Table 5.5 shows, the average recall score is below 1 for all domains other than ROVERS. This is the manifestation of the problem described above, in which goals can be achieved early in the plan, then deleted and re-achieved. Figure 5.20 shows the times[13] at which goal facts are deleted in the IPC test domains. Domains such as DEPOTS retain a high average *recall* score through the volume of goals which are achieved during the 15 problems tested, while others such as STORAGE have a lower *recall* score as there are generally fewer goals achieved during a typical plan (see Figure 5.2, page 142).

[13]Goal deletion times are normalised against their respective plan lengths.

Domain	Average F$_1$	Average Precision	Average Recall
DEPOTS	0.59	0.44	0.96
DRIVERLOG	0.79	0.74	0.88
ROVERS	0.54	0.37	1.00
SATELLITE	0.72	0.58	0.96
ZENOTRAVEL	0.77	0.66	0.95
OPENSTACKS	0.63	0.46	1.00
STORAGE	0.30	0.19	0.80
TRUCKS	0.63	0.46	1.00

Table 5.5: The averaged final P+R and F$_1$ scores for all IPC domains across problems 1–15.

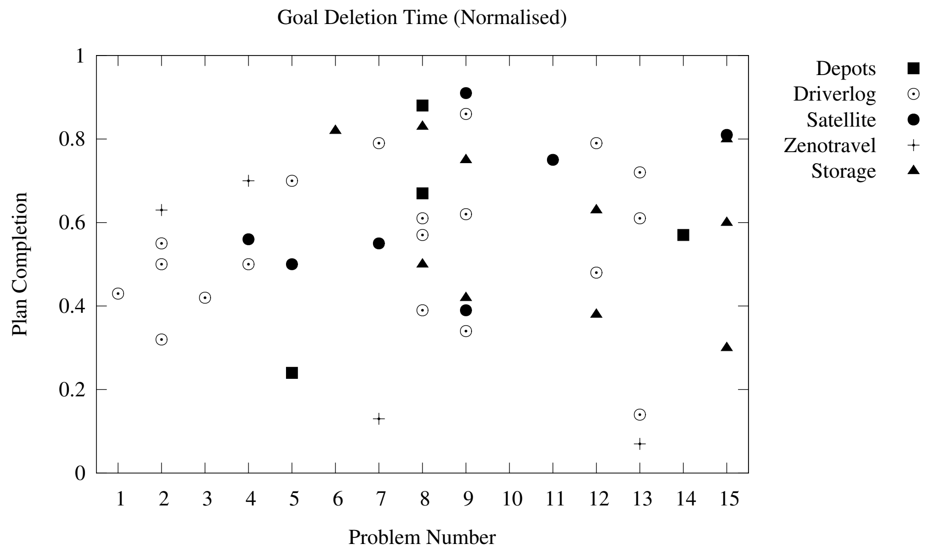


Figure 5.20: The normalised timepoints at which at least one of the agent's true goals is deleted for the plans used in evaluation. ROVERS, OPENSTACKS and TRUCKS are not included as the true goal is always comprised of *strictly terminal* facts.

5.12 Predictive Accuracy

The *intermediate* hypothesis results presented above show how the accuracy of hypotheses change over the duration of observation, using P+R as a means of scoring. However, in this form the results do not provide any context as to the predictive performance of IGRAPH. That is, the above scores may simply be derived from the final *intermediate* hypothesis only. Therefore, it is beneficial to

Configuration	Before	Same	After	None
$\langle h_{max}, W_{ML} \rangle$	10929	1625	2331	3463
$\langle h_{max}, W_{MLT} \rangle$	12764	1822	3109	2480
$\langle h_{max}, W_{SA} \rangle$	18928	3966	1120	1518
$\langle h_{ff}, W_{ML} \rangle$	11762	1624	1991	3574
$\langle h_{ff}, W_{MLT} \rangle$	13580	2376	2426	2601
$\langle h_{ff}, W_{SA} \rangle$	25885	3384	1071	1843
$\langle h_{cea}, W_{ML} \rangle$	11746	1761	1966	3381
$\langle h_{cea}, W_{MLT} \rangle$	13901	2577	2128	2366
$\langle h_{cea}, W_{SA} \rangle$	19964	3374	1007	1693
Total	139459	22509	17149	22919

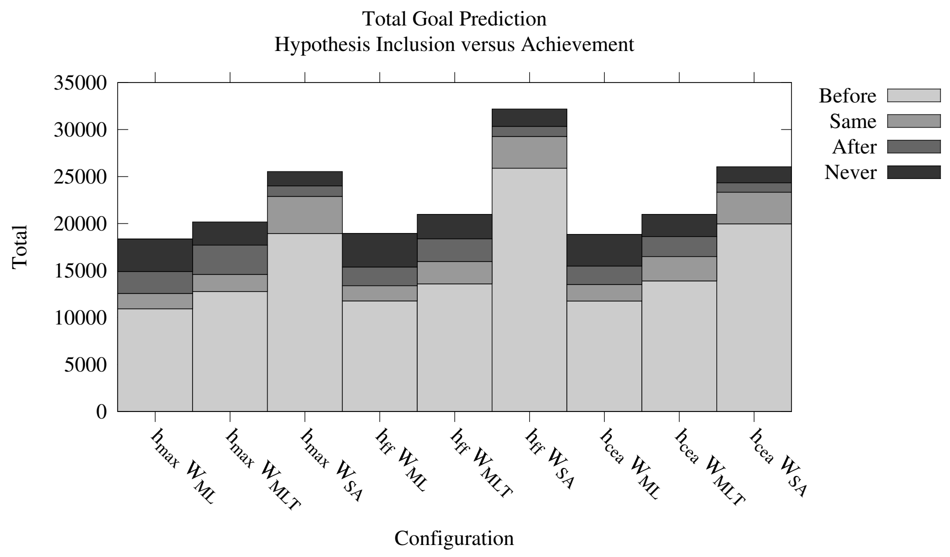
Table 5.6: Specific goal prediction results for each configuration. Configurations using the W_{SA} work function have the highest number of correct goal predictions prior to achievement, as well as the fewest number of post-achievement predictions.

look at when a goal is included in a hypothesis relative to its achievement in the plan. Note that this includes situations where the goal has been added, deleted and then re-added to a state.

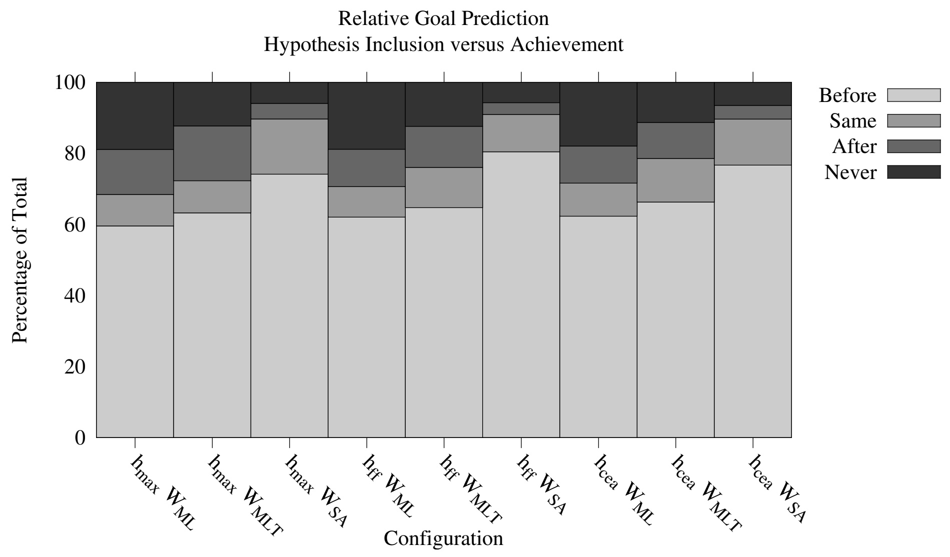
Table 5.6 shows how each configuration of IGRAPH performs at including goals in hypotheses *before*, at the *same* time and *after* achievement of the goal itself by the executing plan. Results for goals which are *never* included in a hypothesis are also given, with each configuration showing the total results for all IPC3 and IPC5 domains tested. The total given for a configuration is the number of times a goal is added to the intermediate hypothesis for the first time.

From these results, configurations using W_{SA} appear to perform the best at predicting the goal before achievement, with the h_{ff} heuristic providing the fewest post-achievement inclusions. Figures 5.21a and 5.21b show these results relative to one another. The variance shown in Figure 5.21a between the total number of goals included in hypotheses is caused by each configuration producing a different number of goals in each hypothesis, and the scoring mechanism given in the previous paragraph. For instance, W_{SA} appears to add true-goals to hypotheses more frequently than other configurations. In all cases the number of goals which are added to a hypothesis before or at the same time as their achievement is greater-than 70%.

With a *before-achievement* prediction rate of approximately 60% over all con-



(a) Total accuracy of goals added to *intermediate* hypotheses.



(b) Relative accuracy of goals added to *intermediate* hypotheses.

Figure 5.21: The *total* and *relative* accuracy of goal predictions produced under each configuration of IGRAPH. As Figure 5.21a shows, configurations using W_{SA} generate more hypotheses which successfully predict the goal. Note that the totals in Figure 5.21a are not equal despite being evaluated on the same domains and problem files. This is due to each configuration adding or removing the relevant goal fact multiple times during observation. That is, configurations using W_{SA} add/re-add goals to hypotheses more often than those using W_{ML} and W_{MLT}, with each addition being counted towards the total score once.

figurations and an *at-achievement* rate of 10–15%, IGRAPH provides accuracy which is sufficient for select applications. For example, in a video game environment, such accuracy may be sufficient to provide an intelligent response to a player's strategies. Alternatively, it may provide useful guidance to an operator on a hacker's behaviour within a company network.

Outwith predictive aspects, the ability to determine the goal *post-achievement* (5–10% accuracy) echoes Hong's earlier work in goal diagnosis [89]. However, Hong's work is most similar to the *final* hypothesis results given previously, in that the system has been told of plan termination. IGRAPH demonstrates that on average 80–90% of goals can be determined prior to being notified of plan termination.

5.13 Plan Length Prediction

As described in Section 3.4.3, in the complete goal-space model it is possible to estimate the remaining number of observations as the heuristic estimate to the most-probable goal. This concept also carries over to the relaxed goal-space model, in which suboptimal heuristics can be used to estimate the number of steps remaining, ε.

After each observation is processed, IGRAPH estimates the number of remaining observations in order that this value can be used to generate ε *bounded hypotheses*. The accuracy of these hypotheses is considered in Section 5.14. However, before this the accuracy of ε itself is considered.

As the value for ε is equal to a heuristic estimate, its accuracy is naturally affected by the heuristic used. Figure 5.22 shows the difference in the estimated number of steps versus the actual number of steps remaining across each configuration on each IPC domain as a Gaussian distribution, clamped to the 95th percentiles. These figures also show the maximum and minimum estimations, which appear as the upper and lower bounds of each candlestick.

The accuracy of the estimates varies greatly across each domain and configuration, with the heuristic used causing the greatest variance (as expected). In Depots and Driverlog, ε varies only slightly between configurations, with all underestimating the number of steps remaining in virtually every test. Rovers and Storage show the best results, with the variance in ε revolving around zero (which would be a correct estimate of the number of steps remaining), although the standard deviation is still ±10–20 steps. In both cases, and more generally, h_{ff} provides the least variance in estimates. This may be due to the fact it is the only heuristic which takes positive interactions of actions into account when producing estimates.

In most of the remaining domains, configurations which use the h_{cea} heuristic are highlighted due to the number of extreme over-estimations. In the case of SATELLITE, OPENSTACKS and TRUCKS these over-estimations can be thousands of steps from the true value, while ROVERS shows the same behaviour at an order of magnitude less. This behaviour is caused by the large number of mutex-sets in these domains, combined with the additive nature of h_{cea}. As hypotheses in IGRAPH often verge towards state-estimations, the number of facts contained in them can be far larger than the true goal. Given that the h_{cea} heuristic assumes that all goals are independently achieved, it simply sums together the estimate of each fact in a hypothesis when in reality only a small subset of the goals will be achieved, or that the achievement of these will be done concurrently. This is both a reflection of the relaxed goal-space model and the heuristic used.

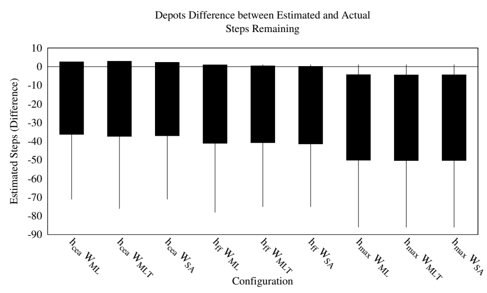
The large variance in estimations within the results is caused by ε being equal to the estimate of the *intermediate hypothesis* produced after the respective observation. That is, $\varepsilon_t = h(Hyp_t)$, where t is the current timestep. As this is simply the maximal grouping across $\mathcal{G}$, its value will change according to the facts chosen. In practice, the observed behaviour is that ε decreases by 1 after each observation for a series of steps, before the *intermediate hypothesis* changes to one which is sufficiently different so as to cause a large change in ε. This behaviour then repeats throughout observation, to a varying degree depending upon the domain and heuristic used.

While the value of ε appears to be unreliable for most domains which do not have an underlying transportation component in their goals, it is also used as an upper-bound in the production of *bounded hypotheses* (Section 3.4.4). The following section will analyse the accuracy of hypotheses produced with ε.

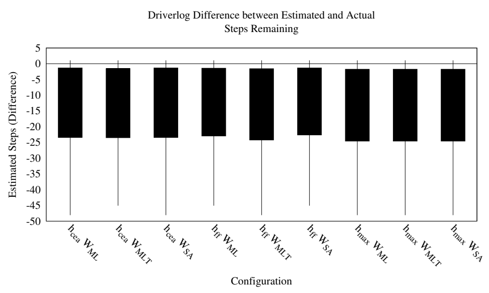
5.14 Bounded Hypothesis Evaluation

Recall that Section 3.4.4 proposed *bounded goal hypotheses* as a means of estimating the agent's goal in n steps. These hypotheses are constructed not to determine the agent's final goal G^*, but rather to predict the intervening steps on the path to the goal[14]. While the estimated number of steps remaining ε has been shown

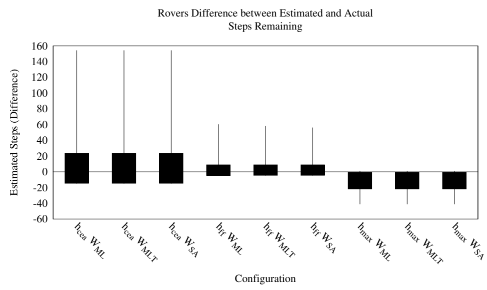
[14]Note that the term "intermediate goal" is intentionally avoided, to prevent confusion with *intermediate hypotheses*, which are only produced after an observation and hypothesise the terminal goal, G^*.



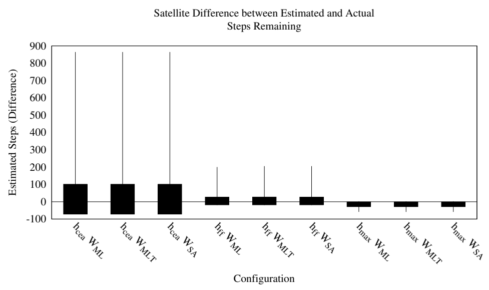
(a) Depots



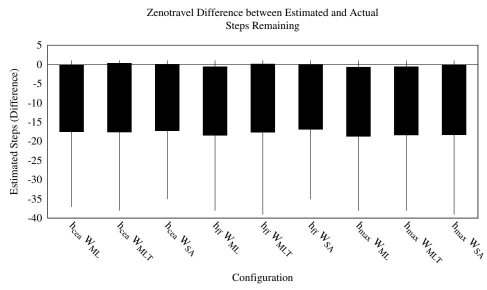
(b) Driverlog



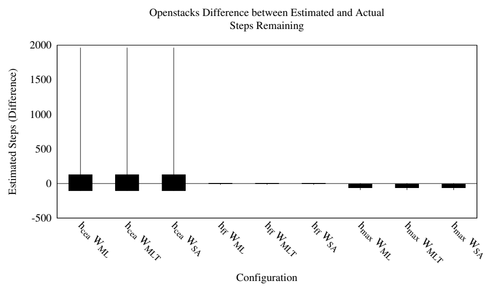
(c) Rovers



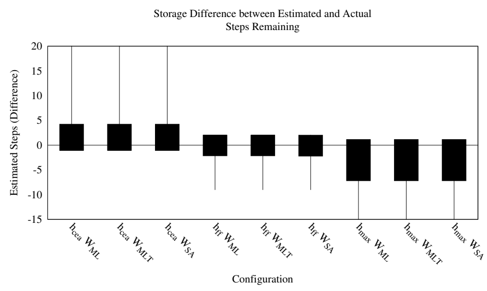
(d) Satellite



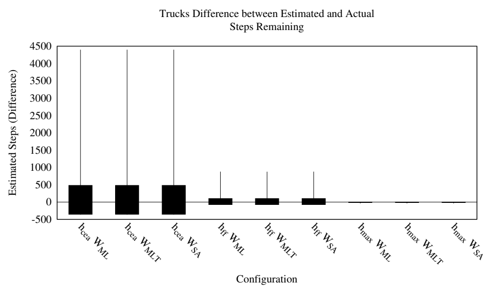
(e) Zenotravel



(f) Openstacks



(g) Storage



(h) Trucks

Figure 5.22: The *difference* in standard deviation of each estimated number of remaining steps, ε, in each domain and configuration. Black bars represent a Gaussian distribution with 95% confidence, with upper and lower bounds represented by lines. Negative values indicate that ε is an under-estimation of the true value, while positive numbers indicate an over-estimation.

to be inaccurate on most domains, it does offer a convenient upper-limit on the number of *bounded hypotheses* which should be produced at each timestep.

After each observation, n bounded hypotheses are produced, where $n = \{x \in \mathbb{Z}^+ | 1 \leq x \leq \varepsilon\}$. A bounded hypothesis, Hyp^b_c has an associated creation, c, and bound time, b. As these are simply standard hypotheses, albeit computed using a slightly more restrictive technique, they can be evaluated in both the context of being a *goal* and *state* hypothesis.

With this said, in practice it makes sense to only consider bounded hypotheses in the context of goals, rather than as states. This is because considering bounded hypotheses as states can lead to inflated P+R scores,due to the gradual nature of state transitions. For example, if the bounded hypothesis for the state following S_1 ($b = 1$) is simply $Hyp^b = S_1$, then the hypothesis will probably have a high P+R score due to the relatively small changes caused by the observed action's add and delete effects. That is to say, if only a small subset of the state changes after each observation, a "good" bound hypothesis is simply equal to the current state.

To account for this potential bias in results, bounded hypotheses are therefore scored against the *difference* between the state in which they were created c, versus that in which they should be true, $c + b$. This is formally defined in Definition 26.

Definition 26. Bounded Hypothesis Score
Given a bounded hypothesis, Hyp^b_c, where $c = \{x \in \mathbb{Z}^+ | 0 \leq x\}$ is the time at which the hypothesis was created and $b = \{x \in \mathbb{Z}^+ | 1 \leq x\}$ is the time at which the hypothesis is expected to be true, the score for the hypothesis uses standard precision and recall, with the exception that the true sub-goal G^*_{c+b}, is equal to $G^*_{c+b} = S_{c+b} \setminus S_c$.

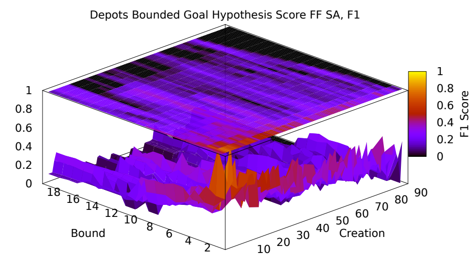
The sub-goal is considered to be the difference between the facts in the bound hypothesis and those in the actual state. For instance, if $S_c = \{A, B, C\}$ and $S_{c+b} = \{A, D, E\}$, then the sub-goal which the *bounded hypothesis* will be compared against is $G_{c+b} = \{A, D, E\} \setminus \{A, B, C\} = \{D, E\}$. That is, the agent was trying to achieve $\{D, E\}$ on the way to achieving the final goal, and that the *bounded hypothesis* should be scored against these facts only.

Figures 5.23 shows typical bounded hypothesis F_1 scores for the domains tested, based on the $\langle h_{ff}, W_{SA}\rangle$ configuration which has been shown to have the best performance in generating intermediate hypotheses. The complete set of results for all configurations are omitted here for brevity, but can be found in full in Appendix B, including *precision* and *recall* results. Note that as stated in Section 5.5, bounds which exceed the end of the actual plan do not have bounded hypotheses generated, as they cannot have a score assigned.

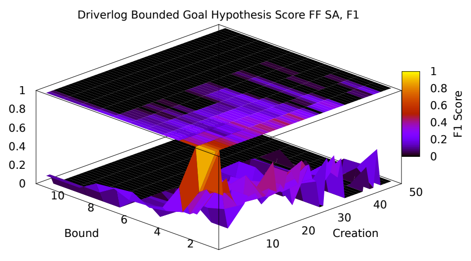
These results echo those of the *intermediate* hypothesis results, in that certain domains perform well, while others do not. Those such as ROVERS, STORAGE and OPENSTACKS receive high scores for bounded hypotheses with both low or high creation and bound times. In the case of OPENSTACKS, determining the true final goal is trivial, as the true goal is always made up exclusively of ST literals, with no extraneous ST facts which are not part of the goal. This leads to very high accuracy results at extreme bounds where the number of facts which can be achieved at this minimum distance is small. STORAGE offers an interesting contrary to this, as goals in this domain are non-ST, yet bounded hypotheses appear to perform well across creation and bound times, with the exception of the plan-horizon boundary where accuracy falls off. This is caused by the recogniser being unable to determine which container a crate will be unloaded into (as there can be multiple crates in a warehouse).

The F_1 results for ROVERS show strange behaviour, in that the bounded hypotheses become worse as observation proceeds. This is caused by the precision of the hypotheses being lowered as creation time progresses. At the start of observation, bounded hypotheses are highly accurate as the domain exhibits a strong causal chain of actions, with largely non-overlapping plan threads. However, as observation proceeds, a large number of reachable ST facts will remain unachieved, but will continue to be put forth as being achievable within a single observation. This same behaviour is echoed in the results for SATELLITE, but at a much lower F_1 score. Appendix B.3 shows that bounded hypotheses in this domain largely receive perfect recall, but very low precision scores.

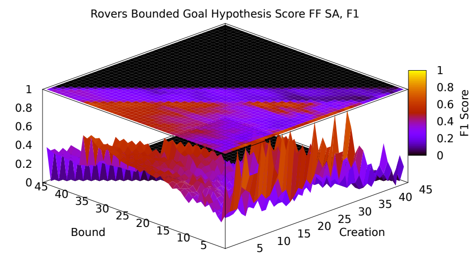
The results for the remaining domains (DEPOTS, DRIVERLOG, ZENOTRAVEL, TRUCKS), are more in line with the expected accuracy of bounded hypotheses. In general, hypotheses with high bound times have low accuracy, while those with low bounds are more accurate. In fact, all domains have a strong accuracy relating to a bound of $b = 1$, regardless of creation time, although this can drop off rapidly depending upon the domain. Of these four domains, the only notable result is that of TRUCKS hypotheses which are close to the end of observation, where a high accuracy is obtained (as the recogniser realised that packages are about to be delivered to their final destination).



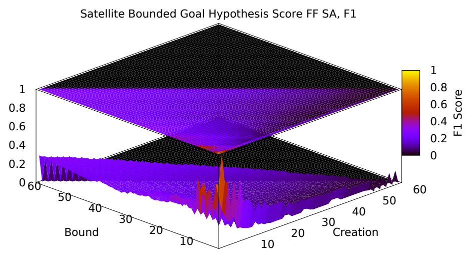
(a) Depots



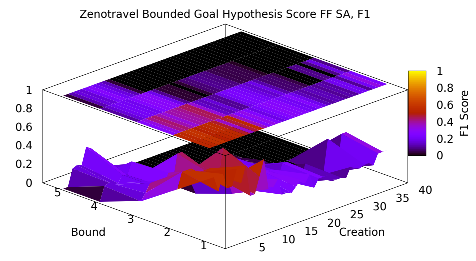
(b) Driverlog



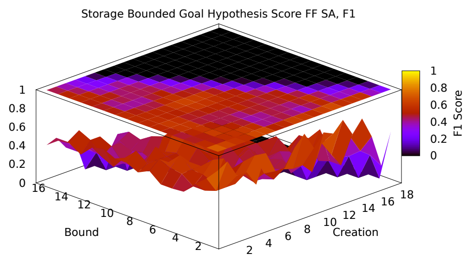
(c) Rovers



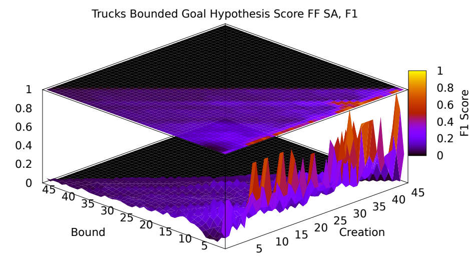
(d) Satellite



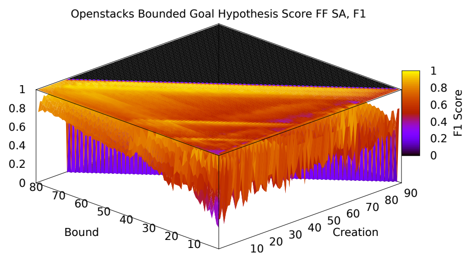
(e) Zenotravel



(f) Storage



(g) Trucks



(h) Openstacks

Figure 5.23: Results for the average F_1 score using the $\langle h_{ff}, W_{SA} \rangle$ configuration on all IPC domains and problems.

These results appear to indicate that generating bounded hypotheses is a viable part of the heuristic model. While these are largely linked to underlying domain structure (as would appear to be the general case also), they do offer use in certain scenarios. In particular, applications in which the observed must be intercepted *prior* to the termination of their plan could benefit from such hypotheses. However, the uncertainty and high branching factor of some domains means that some form of probabilistic planning, such as a Markov decision process [15], would be helpful in filtering out facts which are unlikely to appear in most plans at a given bound.

5.15 Impact of Domain Analysis on the Initial Probability Distribution

In Section 4.4, various domain analysis techniques were presented to enable the creation of an intelligent initial probability distribution, prior to observation beginning. This involved computing the probability of a standard literal being a goal given its *causality value* and its associated locations in the causal graph. This section will evaluate how successful the use of this initial distribution is, when compared against a simple *uniform initial distribution.* For clarity, the initial distribution produced using domain analysis shall be referred to as the *contextualised* initial distribution.

The initial probability distribution is useful in two regards. Firstly, it naturally offers the potential for a "head-start" in the recognition process, enabling more accurate hypotheses to be produced early in observation. Secondly, it allows the construction of a hypothesis prior to observation beginning, rather than the random hypothesis which would be produced using a uniform distribution.

5.15.1 Initial Hypothesis Accuracy

The first beneficiary of domain analysis is naturally the *initial hypothesis.* This is constructed prior to recognition beginning and uses only the initial state; grounded problem representation and any information extracted through the domain analysis to select facts in $\mathcal{G}$ which are the best goal candidates.

IGRAPH generates the initial hypothesis in the same manner as *intermediate* hypotheses (page 77). Figure 5.24 shows the increase in F_1 score when the contextualised initial distribution is used over the uniform distribution. Table 5.7 expands upon this by giving details of the minimum, maximum and average

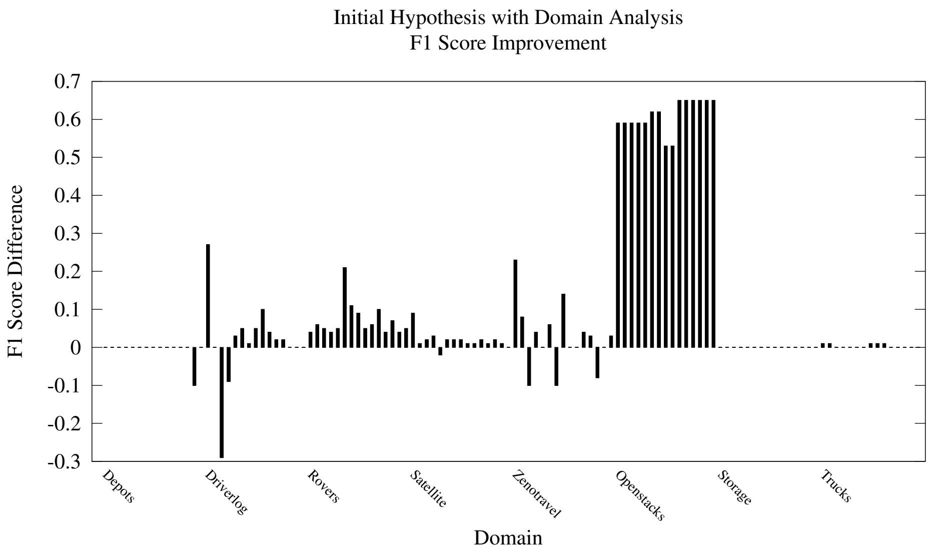


Figure 5.24: The increase in the F_1 score of the initial hypothesis when domain analysis is used to produce a non-uniform initial distribution across the goal-space.

Domain	Total F_1 Increase	Average	Min	Max
Depots	-0.10	-0.01	-0.10	0.00
Driverlog	0.21	0.01	-0.29	0.27
Rovers	1.06	0.07	0.04	0.21
Satellite	0.27	0.02	-0.02	0.09
Zenotravel	0.37	0.02	-0.10	0.23
Openstacks	9.15	0.61	0.53	0.65
Storage	0.00	0.00	0.00	0.00
Trucks	0.05	0.00	0.00	0.01

Table 5.7: Various statistics relating to the improvement in F_1 score for initial hypotheses in the IPC domains. "Total F_1 Increase" is the total *improvement* in the initial hypotheses for a particular domain across all 15 test problems.

improvement in F_1 score, as well as the total improvement across each of the 15 problems associated with a domain.

The results in this table show that the contextualised initial hypothesis benefits only some of the domains tested. Openstacks clearly gains value from the non-uniform initial distribution, with an average increase in the F_1 score of 0.61, while Rovers has a lesser but notable increase. This is again caused by SA facts being present in these domains, and being assigned higher probabilities than mutex goals. Satellite shows minor improvements, however Driverlog and Zenotravel show an increase in accuracy on some domains and a decrease in others. This is most likely caused by the contextualised distribution being unable to determine where a package or passenger will located at the end of the plan.

Depots, Storage and Trucks show no improvement in hypothesis score over a uniform distribution. This may be partly explained by the hypothesis extraction algorithm always preferring to select a true literal rather than an *all false* goal, as the contextualised distribution often also assigns these a higher initial probability. However, these domains are also simply hard domains on which to perform recognition. For example, without any observations it is impossible to determine which location a crate will be delivered to, or which surface it will be stacked onto.

5.15.2 Intermediate Hypotheses

As with previous sections, the impact of the initial distribution is evaluated against the intermediate hypotheses produced by IGRAPH. Figure 5.25 shows the percentage increase in the total intermediate F_1 scores when using the contextualised initial distribution rather than the uniform distribution, across all domains and all configurations. This indicates that on average there is a slight improvement in overall score, but that there are multiple cases where this score decreases instead.

Table 5.8 shows the average percentage increase in the total F_1 score across problems 1-15 for each domain. This shows that configurations which use the W_{SA} work function have the highest average increase, with more accurate heuristics gaining the most from the contextualised distribution. This is in line with previous results involving *intermediate* hypotheses.

The results show that by using the contextualised initial distribution, most domains benefit from an increase in total intermediate hypothesis accuracy. Figure 5.26 shows these results on a per-domain and per-configuration basis. Rovers, Satellite, Zenotravel, Openstacks and Storage show the most consistent improvements across all heuristics and work functions. This could be a result of these domains exhibiting a strong *causal hierarchy*, with goals often based on variables associated with the leaves of the causal graph.

Results for some domains such as Trucks are more configuration-dependent. In this case, configurations using W_{SA} show a reduction in overall accuracy, while

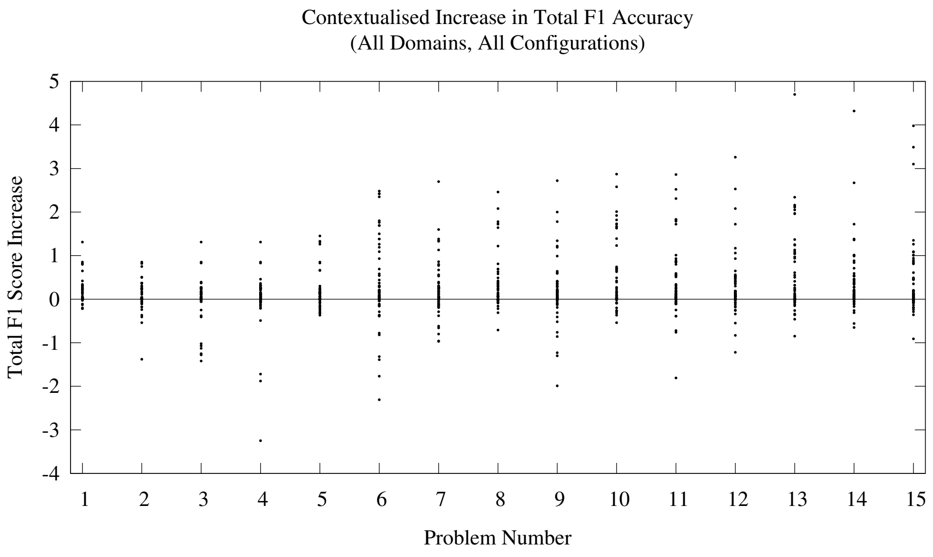


Figure 5.25: A point-cloud representation of the overall increase in total F_1 score for every problem and configuration.

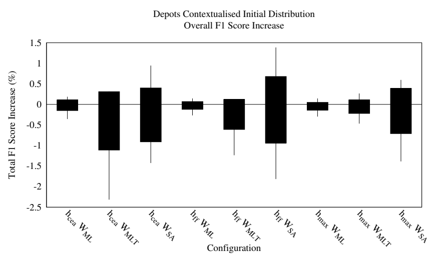
Domain	**Configuration**								
		h_{max}			h_{ff}			h_{cea}	
	W_{ML}	W_{MLT}	W_{SA}	W_{ML}	W_{MLT}	W_{SA}	W_{ML}	W_{MLT}	W_{SA}
Depots	-0.04	-0.05	-0.16	-0.02	-0.40	-0.26	-0.03	-0.25	-0.13
Driverlog	0.00	0.05	-0.24	0.01	0.11	-0.48	0.09	**0.13**	-0.20
Rovers	0.48	0.53	**1.12**	0.47	0.59	1.09	0.47	0.57	1.08
Satellite	**0.27**	0.05	0.15	0.07	0.05	0.16	0.06	0.05	0.22
Zenotravel	0.07	0.11	**0.84**	0.09	0.05	0.59	0.07	0.02	0.71
Openstacks	0.18	0.13	1.59	0.07	0.00	1.74	0.16	0.01	**2.46**
Storage	0.03	0.27	0.38	0.05	0.40	0.30	0.09	0.32	**0.50**
Trucks	0.06	0.05	-0.35	0.01	0.01	-0.12	0.02	0.02	-0.17
Total	1.04	1.14	3.33	0.76	0.81	3.04	0.94	0.87	**4.46**

Table 5.8: The average percentage increase in overall intermediate hypothesis F_1 score when the *contextualised* initial distribution is used over a uniform distribution. The highest non-trivial increase in scores are highlighted.

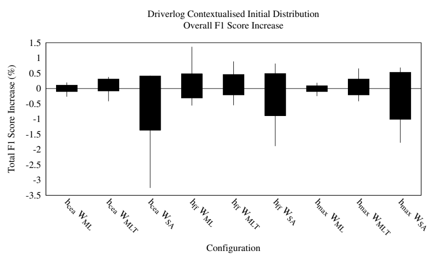
others show a trivial increase. This relates to the extremely low precision in Trucks hypotheses, which will cause a minor increase in overall accuracy due to the low corresponding F_1 scores. As discussed above, Depots and Driverlog show no overall increase in total score, regardless of configuration used.

The results appear to indicate that small increases in total F_1 score are possible when basic domain analysis is used. However, this is largely dependent upon the underlying domain structure.

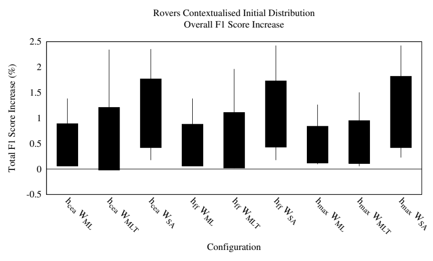
Of the domains which do show an improvement in the initial hypothesis or



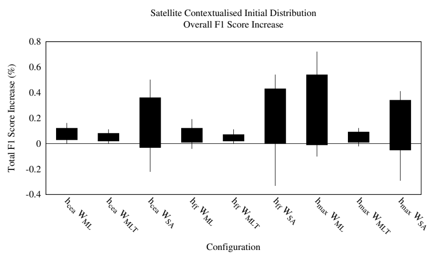
(a) Depots



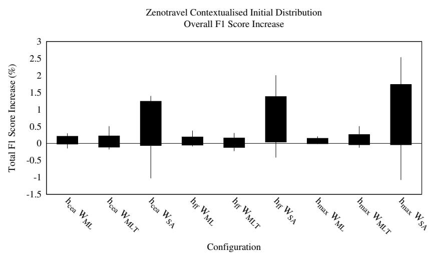
(b) Driverlog



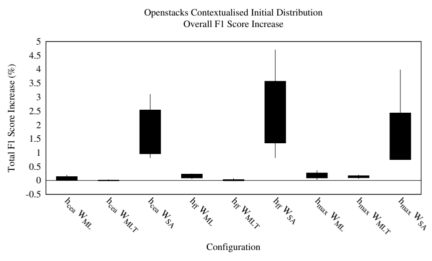
(c) Rovers



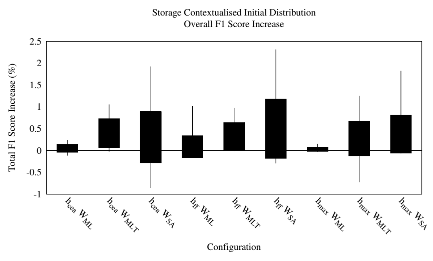
(d) Satellite



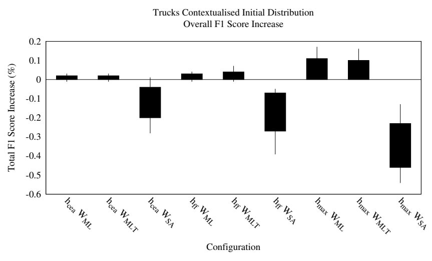
(e) Zenotravel



(f) Openstacks



(g) Storage



(h) Trucks

Figure 5.26: A domain-specific visualisation of the difference using a contextualised initial distribution makes to the total F_1 score. Boxes are centred around the mean difference in F_1 result (shown in percent) for problems 1–15, with their high dependence upon the corresponding standard deviation. The minimum and maximum difference in F_1 score is shown by the upper or lower point respectively. The x-axis in each graph corresponds to a different heuristic and work function configuration of IGRAPH.

overall F_1 score, this increase is small enough to question how useful the contextualised distribution is once observations have started being processed. It would appear that the first few observations destroy the contextualised distribution, which has the possibility of being more accurate in determining the final goal than the resulting distribution after actions have been observed.

While these results are heavily dependent upon the structure of the underlying state, goal and plan-spaces, many classical recognition applications would be suitable candidates for such analysis. For example, Blaylock's MONROE corpus [18], which models emergency response scenarios, has many strictly terminal facts which could be used to improve the initial probability distribution as above[15].

5.16 Plan/Goal Abandonment

Section 3.6 described how plan/goal abandonment is implicitly present in the heuristic-based model, something that is in contrast to previous work which must explicitly model this [61]. This section shows how abandonment is detected within various domains and at different points of plan observation.

For the purposes of demonstrating abandonment, the IPC domains used elsewhere are put to one side in preference of a simple grid based world. The benefits of this are two-fold. Firstly, they allow the evaluation of IGRAPH on a domain where there is only a single goal (for the agent to be at a location), so no conflicting goals can be included in a hypothesis. Secondly, it simplifies the visualisation of how important goal abandonment speed is. That is, whether it is harder to detect a goal being abandoned at 10% or 90% of the observed plan. The cost of this grid based model is that the F_1 score is not a useful metric, as there is often a large number of goals which become closer at the same time, making differentiation of the true goal difficult. Therefore, the *probability* associated with the goal at each observation is reported instead[16].

The plans used in this section are generated by modifying the JavaFF planner [44] to produce a chain of plans which have their goal switched at a pre determined percentage of the overall plan. For instance, if goals are to switch after 30% of observations, a plan is generated for the first randomly selected goal, then

[15]An attempt was made to automatically derive a PDDL representation of the MONROE domain using only the action traces contained within the corpus. Unfortunately, the number of action templates present in the plans proved too complex for the tool used (LOCM2 [45]).

[16]In the case of a simple grid world, using the probability as a metric is reasonable, as the relaxed goal-space is equal to the complete goal-space $\mathbb{G}$, meaning the mutex-set restrictions reported in Section 4.3.1 do not apply.

trimmed to include only the first 30% of actions, before another goal is randomly selected and the process repeats. Goals are constrained to be a minimum of 5 plan-steps apart in order to differentiate between abandonment and slightly suboptimal plans[17]. Note that the results presented do not consider whether the goal was correctly hypothesised — merely that the probability of the latter goal exceeds that of the former post-abandonment. This is because the previous sections have demonstrated the recognition ability of IGRAPH, while here the matter of whether a heuristic-based model can detect goal switches is of greater interest.

As an initial example, Figure 5.27 shows a 21x21 grid world where the agent starts in the centre and must move to the South-West corner to achieve their goal. However, this goal is abandoned at 10%, 50% and 90% of plan completion, at which time the agent switches to the North-West, North-East or South-East goal respectively. Figure 5.28 shows the corresponding probabilities of the abandoned and new goals over plan observation. For this simple world, IGRAPH is initialised to use the $\langle h_{max}, W_{SA} \rangle$ as estimates will be optimal regardless of heuristic choice, while a uniform initial distribution is applied. In all cases the goal is correctly recognised, although later goal abandonment naturally leads to a longer delay in detection.

While the IPC domains are suitable for standard recognition, they are a poor fit for demonstrating abandonment detection. Therefore, a simple domain in which the agent moves around a dense but loosely connected city environment is used, where being at a location is the only possible goal. Figure 5.29 shows a complete enumeration of the state-space for such a domain with 1000 possible locations. Local areas of the state-space are clearly visible, along with paths between some of these local problems. These are intended to mimic areas of a more complex state-space in which local neighbourhoods can be solved through multiple means. Some of these neighbourhoods are visible as outlying points in Figure 5.29b, while the central core shows highly-connected areas of the state-space. Once the agent enters one of these neighbourhoods the recogniser should rapidly converge onto those goals which are in this local area.

Figures 5.30 and 5.31 shows the probability of each abandoned goal when 1, 4 and 9 goals are abandoned throughout plan execution (plus a further goal

[17]It is of course possible for an agent to abandon their goal on the last plan step, but under these circumstances a heuristic-based recogniser would most likely be unable to determine that this was the case, and not simply that the new goal was the true goal all along.

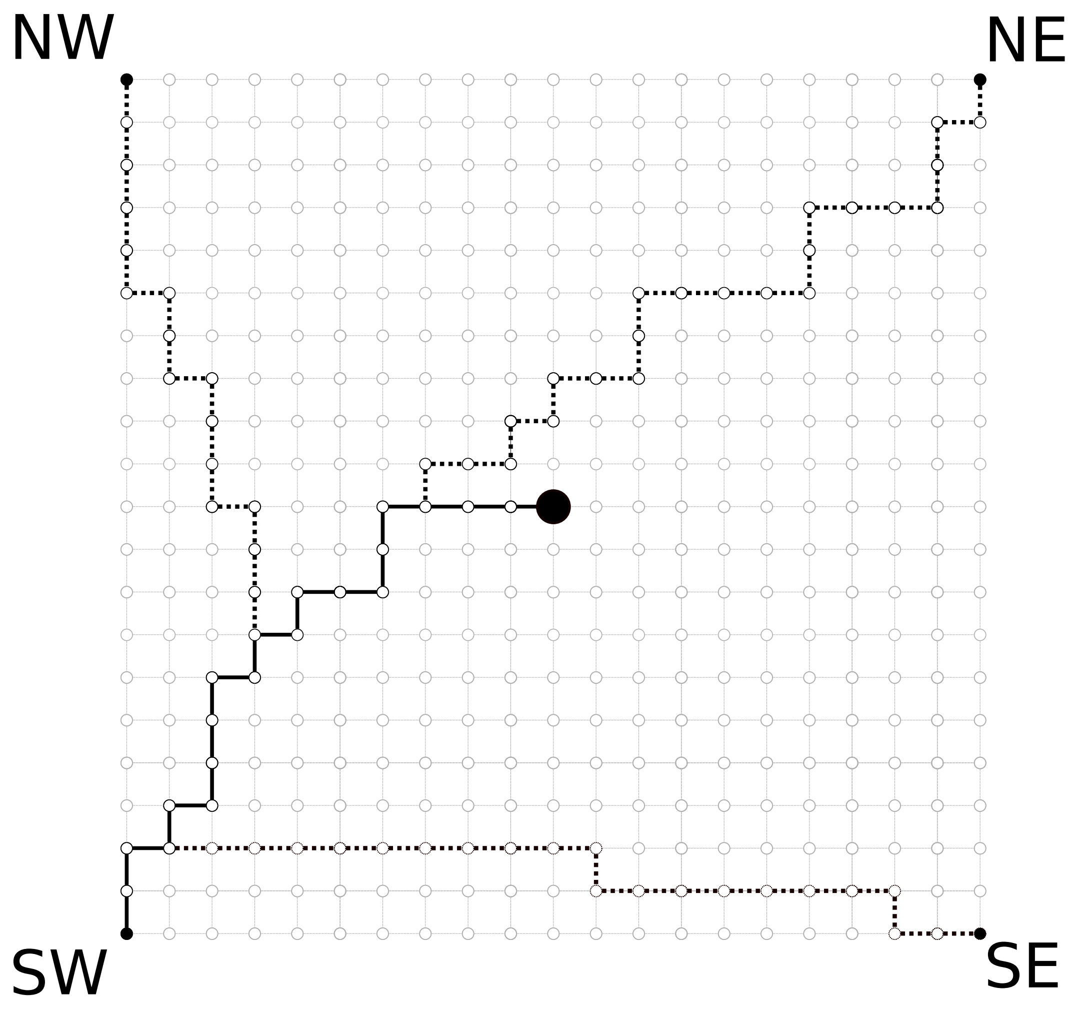
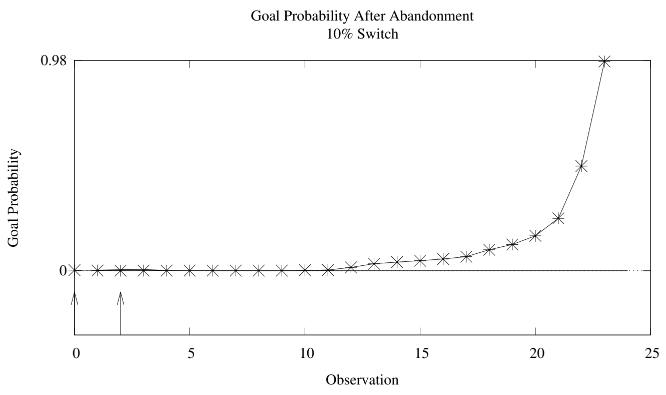
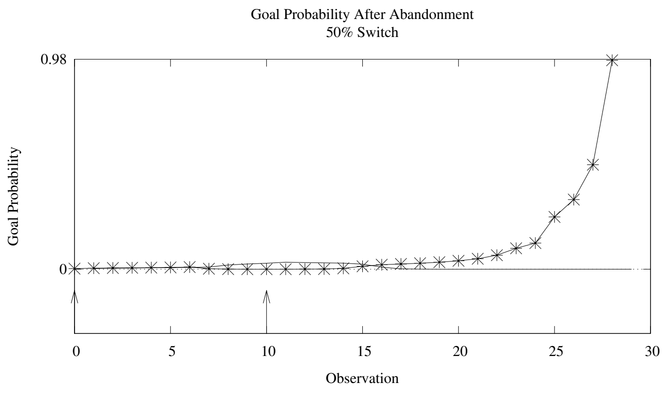


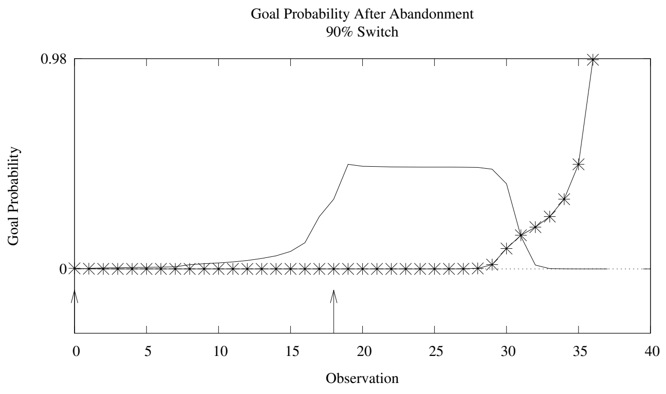
Figure 5.27: Goal abandonment in a 21x21 grid world where the agent starts in the centre and the initial goal is always to reach the South-West corner. The same initial plan is used in all cases, as marked by a solid line. Plans generated after abandonment are shown by dotted lines. The North-East goal is switched to after 10% of the initial plan has been executed; the North-West goal after 50%, and the South-East goal after 90%. Other results (such as switching to the North-West goal at 10% or 90%) are omitted as any differences in results are minimal.



(a) Goal abandoned at 10% of observation for NE goal.

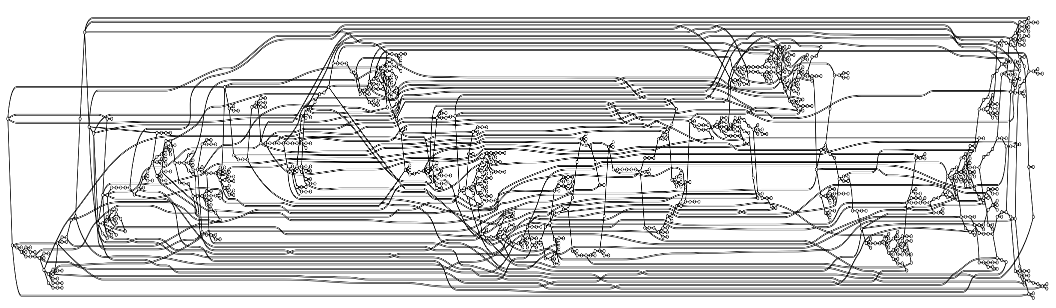


(b) Goal abandoned at 50% of observation for NW goal.



(c) Goal abandoned at 90% of observation for SE goal.

Figure 5.28: Examples of an agent in a grid world domain abandoning the current goal at different stages of plan completion, using h_{max} in all cases, and W_{SA} as the work function. Arrows along the x-axis indicate the time at which the current goal was abandoned, with the true goal probability being indicated as a line with star points.


(a) A standard map of the city state-space.

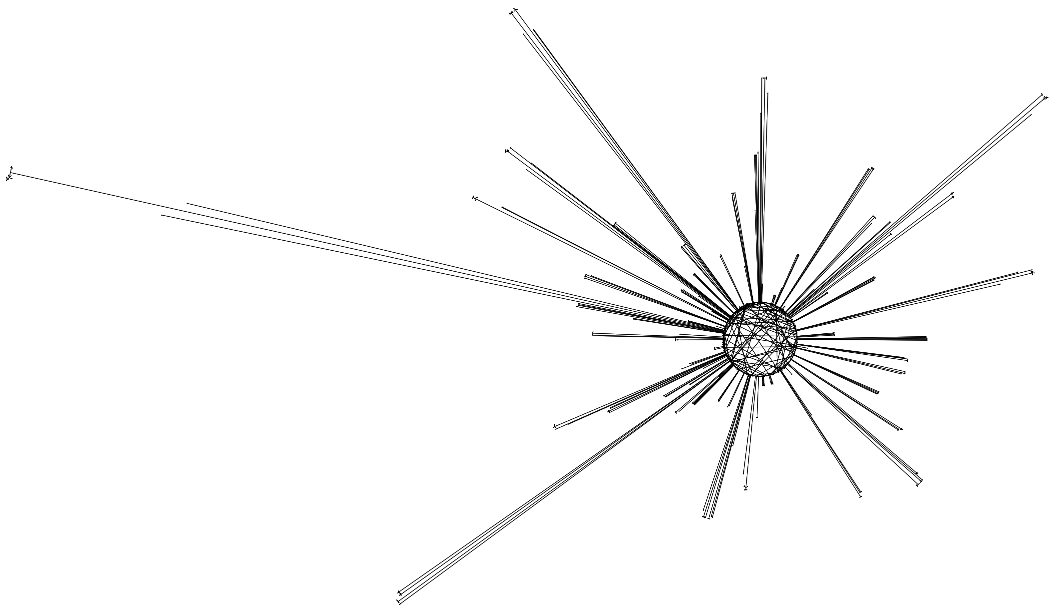
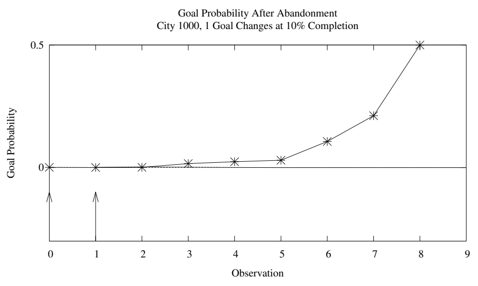

(b) A circular map of the city state-space which highlights a (relatively) densely connected central path network and outlying local problems.

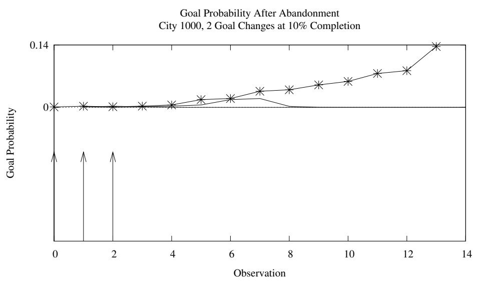
Figure 5.29: Two representations of a 1000 node city state-space. The first figure shows a central cluster of paths through the state-space, with various edges between local problems visible. The second shows the same space with densely connected nodes clustered in the centre of the circular area. Outlying local areas of the state-space are highlighted as arms emanating from the centre.

indicating the agent's final position). These results are again generated using the $\langle h_{max}, W_{SA} \rangle$ configuration. The W_{ML} and W_{MLT} functions are a weak choice for problems in which goals are expected to be abandoned, as they consider the entire plan trace and assume optimality in either a linear or concurrent form. Instead, W_{SA} only considers the current observation with no historical context. Note that in all cases larger abandonment goal lists are a superset of smaller goal lists.

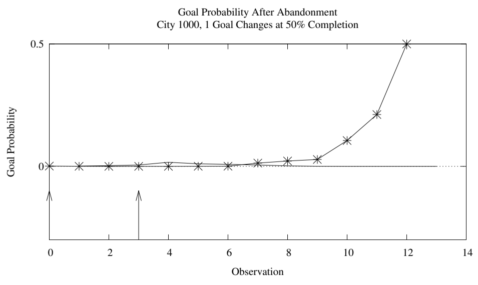
Figure 5.30 shows that goal switches are correctly detected in this movement-



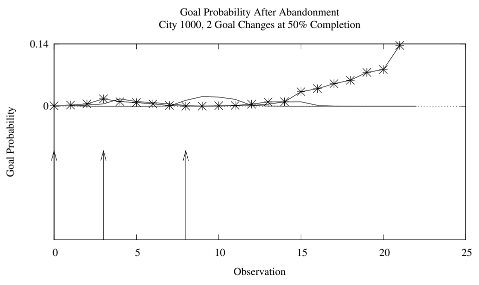
(a) 1 goal abandoned at 10% of plan completion.



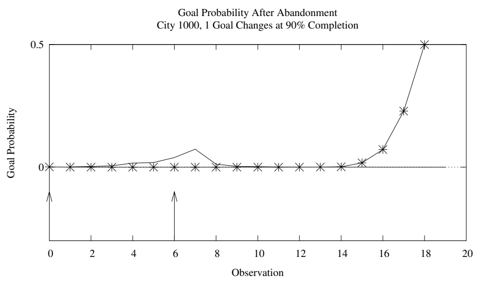
(b) 2 goals abandoned at 10% of plan completion.



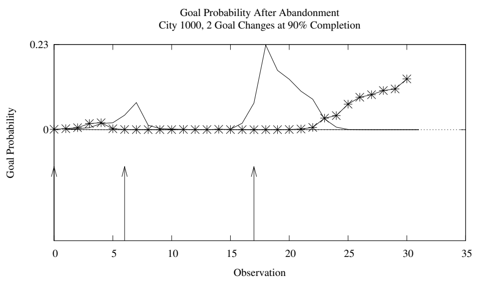
(c) 1 goal abandoned at 50% of plan completion.



(d) 2 goals abandoned at 50% of plan completion.

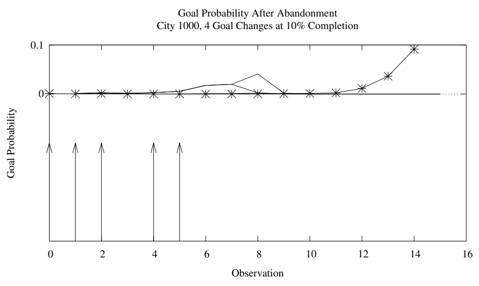


(e) 1 goal abandoned at 90% of plan completion.

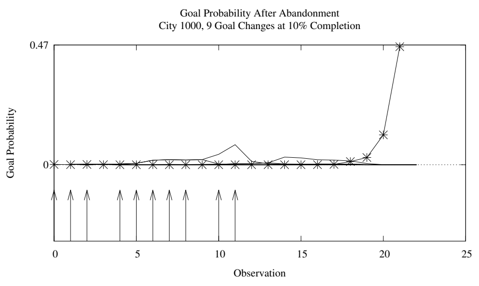


(f) 2 goals abandoned at 90% of plan completion.

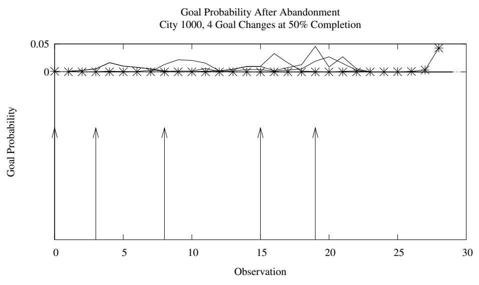
Figure 5.30: Results for abandonment on the city domain for 1 and 2 goals being abandoned at 10%, 50% and 90% of plan completion. Arrows along the x-axis indicate the time at which the current goal was abandoned, with the true goal probability being indicated as a line with star points.



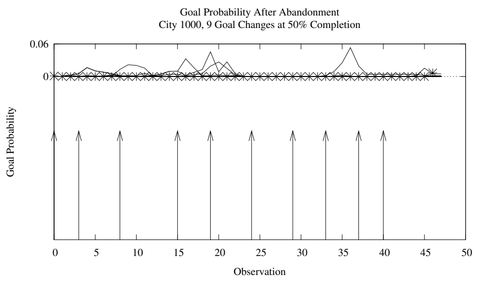
(a) 4 goals abandoned at 10% of plan completion.



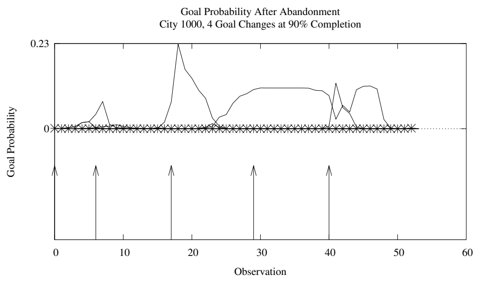
(b) 9 goals abandoned at 10% of plan completion.



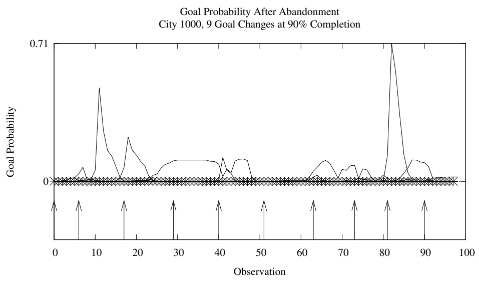
(c) 4 goals abandoned at 50% of plan completion.



(d) 9 goals abandoned at 50% of plan completion.



(e) 4 goals abandoned at 90% of plan completion.



(f) 9 goals abandoned at 90% of plan completion.

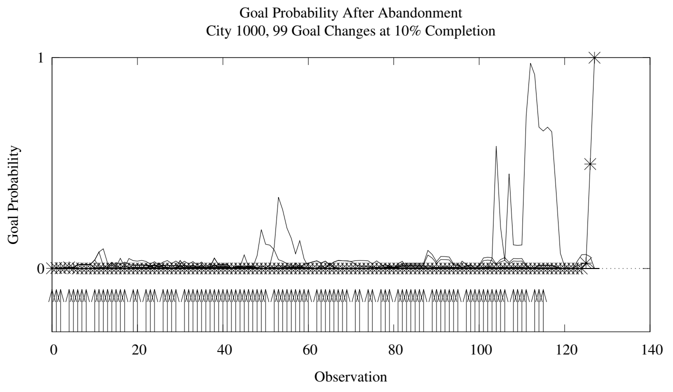
Figure 5.31: Results for abandonment on the city domain for 4 and 9 goals being abandoned at 10%, 50% and 90% of plan completion. Arrows along the x-axis indicate the time at which the current goal was abandoned, with the true goal probability being indicated as a line with star points.

based domain. In particular, Figure 5.30e shows a distinctive dip in the original goal's probability following abandonment at step 6, something which is also visible in Figure 5.30f at both abandonments.

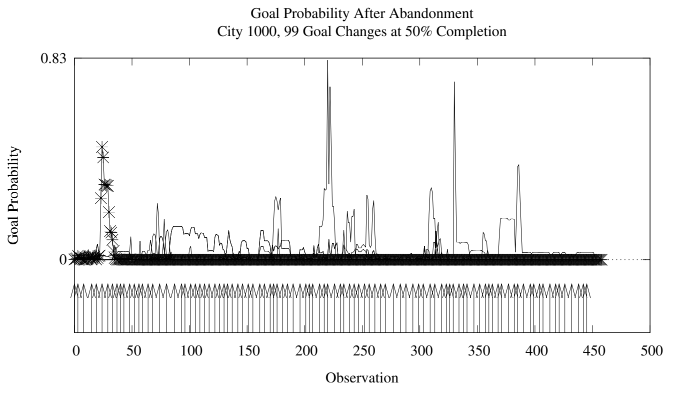
As the number of abandoned goals increases in Figure 5.31 it becomes harder to differentiate between prior and current movement towards a latter goal. For instance, Figure 5.31b shows that the true final goal is not converged upon until the last few observations, with a prior, already abandoned goal having a higher probability from steps 11 – 18. Later plan abandonment as shown in Figure 5.31f makes distinguishing the true goal more difficult still.

The natural conclusion of this is that it will be very difficult to generate an accurate hypothesis for an agent which constantly abandons its current goal. Figure 5.32 shows this situation, where the agent abandons 99 goals before finally terminating their plan. In the case of 90% of plan completion before abandonment (Figure 5.32c), this results in a plan which approaches 1000 steps. By the end of such a long period of observation, virtually all (abandoned) goals have a trivial probability, until the last few observations when the probability of the true goal rises. In this case, the final spike of probability is likely due to the agent entering a sparsely populated neighbourhood of the state-space, where virtually all other members of the goal-space are constantly becoming further away. In fact, the hypotheses output by IGRAPH begin to converge upon an *empty* hypothesis as the number of abandoned goals increases. That is, the *all false* goal becomes the most likely candidate, indicating that the agent has no true goal.

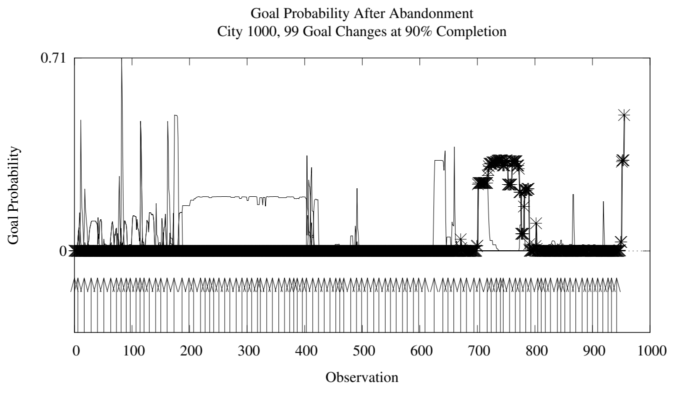
While not the core focus of the heuristic-based model, the ability to detect goal abandonment without additional complexity is useful. In many of the domains where the model is applicable (network intrusion, assistive-agents, video games etc.), subjects often abandon goals in favour of others. With a traditional model of recognition, this would again require that the plan-library contains plans for these abandoned goals, with the added complexity that these must take into consideration the agent's state when the goal is abandoned (as this becomes the initial state of a subsequent plan). The results show that the heuristic-based model may have useful application in such domains, without the overhead required by a model such as that of Geib and Goldman [61]. However, it may be that modifying the Bayesian likelihood function to give higher weightings to more recent observations would help overcome the problem of probabilities decaying over time, in a method similar to reinforcement learning [153].



(a) Goals abandoned at 10% plan completion.



(b) Goals abandoned at 50% plan completion.



(c) Goals abandoned at 90% plan completion.

Figure 5.32: Examples of goal abandonment in the 1000 node city domain across all abandoned candidate goals, where the agent aborts their goal 99 times before plan termination. For ease of reading, only the final goal is highlighted in the legend. At such a high number of goal abandonments it becomes increasingly hard to determine the current goal being pursued, and IGRAPH generates *empty* hypotheses — indicating that it believes there is no active goal for the plan.

5.17 Performance of Plans Generated with Domain Knowledge

In all of the preceding sections, the observer and agent have used *domain-independent* heuristics to derive estimates and plans. However, in both recognition and planning, having access to a heuristic which encodes domain knowledge should allow for better decision-making, lowering plan length and decreasing suboptimal action choices. This section explores whether an improvement in recognition accuracy is possible when the observed agent is known to have *domain-specific* information at their disposal.

Specifically, the tests performed will focus on how IGRAPH performs on plans generated with TLPLAN [7], versus the standard FF planner [88]. TLPLAN uses additional domain knowledge provided by human experts prior to planning beginning, while FF uses the standard domain-independent heuristic described in Section 4.3.3. Both planners competed in the 3rd International Planning Competition [117] where TLPLAN was placed joint first, and FF was placed third. Note that as TLPLAN only competed in IPC3, the IPC5 domains are excluded due to there being no domain-dependent configuration files available.

Whereas previous tests have used the total F_1 score as an indicator of the accuracy of a configuration, this is not possible when differing plan sources are used. This is because a longer plan can result in a higher score due to the presence of additional hypotheses being included. That is, a plan with many low scoring hypotheses could out-score a shorter plan with a few high accuracy hypotheses. However, these issues can be somewhat mitigated by using measurements taken at respective times during execution. For example, the score at 50% of plan observation of a more optimal plan should be higher than the same score for a longer plan[18].

Table 5.9 shows the improvement in plan length for solutions produced by TLPLAN versus FF. In DEPOTS solutions are, on-average, 11.06% shorter, while DRIVERLOG shows no notable difference, and all other IPC3 domains have longer solutions despite the use of control-rules. Figure 5.33 shows a scatter plot of all individual plan lengths for each planner.

Figure 5.34 shows a stacked view of the improvement in F_1 score when TLPLAN sources are used in preference to FF plans. Results above the x-axis indicate that

[18]This assumes that the time to achieve each goal is proportionally greater on the longer plan, which is not always the case, but is a useful metric nonetheless.

Domain	TLPLAN Improvement
DEPOTS	11.06
DRIVERLOG	0.23
ROVERS	-9.66
SATELLITE	-3.96
ZENOTRAVEL	-6.47

Table 5.9: The average improvement in solution length when TLPLAN is used, relative to plans produced by FF.

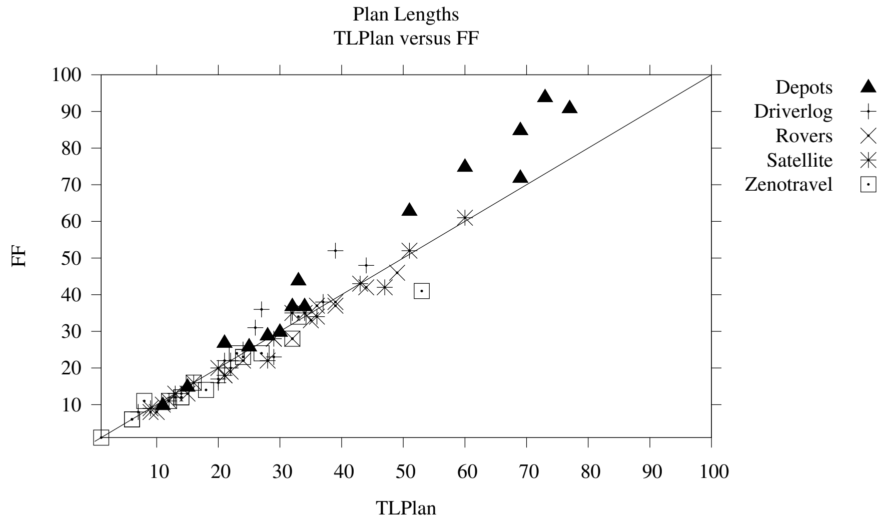


Figure 5.33: A scatter-plot of plan-lengths created by TLPLAN and FF.

the hypothesis generated was better when a TLPLAN source was used over the respective FF plan. At first glance, these results show that any improvement in recognition quality is ambiguous, and appears unrelated to configuration. However, while a configuration or domain-wide improvement is not apparent, problems which *do* show an notable improvement/reduction in score are generally consistent across all configurations. In particular, most ZENOTRAVEL problems have an improvement when using TLPLAN, while DRIVERLOG problems show an almost universal decrease in precision.

More formally, Table 5.10 shows the correlation coefficients between the improvement in plan length when TLPLAN versus FF plan sources are used, and the average increase in respective F_1 scores across each problem and configuration. These are derived from the Pearson linear correlation, and again indicate

Configuration	Depots	Driverlog	Rovers	Satellite	Zenotravel
$\langle h_{max}, W_{ML} \rangle$	0.06	0.03	0.09	-0.61	0.05
$\langle h_{max}, W_{MLT} \rangle$	0.32	-0.01	-0.14	-0.49	-0.05
$\langle h_{max}, W_{SA} \rangle$	0.13	-0.07	0.13	0.23	0.07
$\langle h_{ff}, W_{ML} \rangle$	0.08	0.12	0.04	-0.41	0.08
$\langle h_{ff}, W_{MLT} \rangle$	0.02	0.08	-0.13	-0.49	-0.07
$\langle h_{ff}, W_{SA} \rangle$	0.14	0.11	0.09	0.11	0.11
$\langle h_{cea}, W_{ML} \rangle$	0.12	0.02	0.15	-0.40	0.12
$\langle h_{cea}, W_{MLT} \rangle$	0.11	0.05	0.16	-0.49	-0.02
$\langle h_{cea}, W_{SA} \rangle$	-0.21	0.25	0.24	-0.01	-0.15

Table 5.10: Correlation coefficients of the improvement in plan length when TLPlan plans are used over FF plans, versus specific configurations on each domain tested.

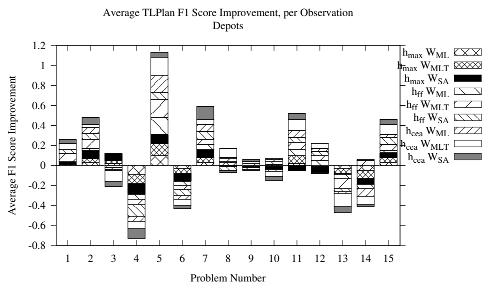
that there is no overall correlation between a reduction in plan length and improvement in recognition score.

Results for Satellite show a correlation coefficient of -0.49 on all configurations which use the W_{MLT} work function, indicating that plan length plays a partial role in scoring. The highest correlation coefficient of -0.61 is also in Satellite, for the $\langle h_{max}, W_{ML} \rangle$ pair, which may be linked to some of the more general reasons for the performance of h_{max} outlined in Section 5.19. However, the actual difference in average F_1 score is trivial, regardless of configuration applied, making this a somewhat misleading statistic.

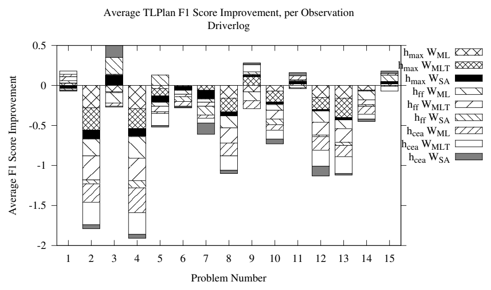
The results for Depots offer the most interesting output, as they do *not* show the expected improvement in recognition score. This is despite plans produced by TLPlan being an average of 11.06% shorter than the FF equivalent.

A natural follow-up to this confusing behaviour is the question of whether performance can be improved if the observer has access to the same heuristic as the agent. In this case, the question is perhaps more succinctly posed as whether doing this can inflict *bias* upon the results. That is, if the observer and agent both make decisions based upon the same information, it is conceivable that the observer will have an unfair advantage in estimating the agent's goal. Such behaviour may explain why a longer FF-derived plan can outscore a shorter TLPLan-derived plan.

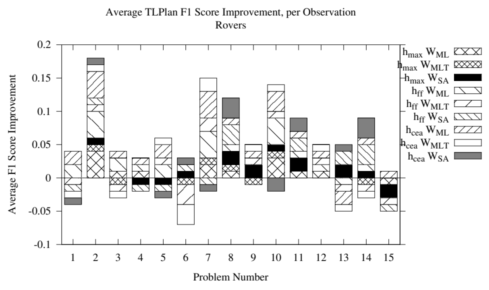
Recall that in the previous IPC tests, the plans which IGRAPH has been evaluated on have come from the best known solutions produced in the associated



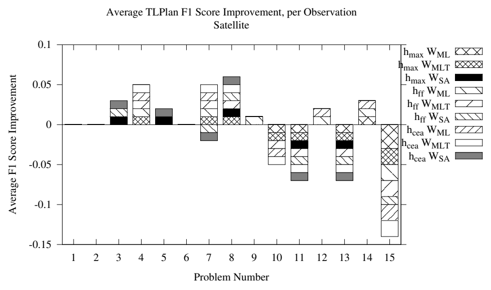
(a) Depots



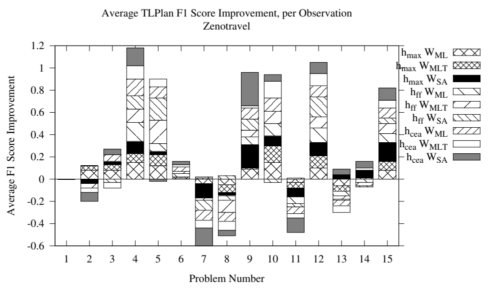
(b) Driverlog



(c) Rovers



(d) Satellite



(e) Zenotravel

Figure 5.34: The average improvement in F_1 score, per observation, when plans generated by the domain-dependent planner TLPlan are used, relative to those generated by the domain-independent planner FF. These averages are taken from the F_1 score at 25%, 50%, 75% and 100% of the respective plan's execution, in order to allow plans of differing lengths to be compared.

competition. For the IPC3 domains, the FF planner was extremely successful, and as such many solutions produced by this planner have been used in testing. Given that the FF *heuristic* has been implemented in IGRAPH, there may be a bias towards plans also produced with this. The following section shall investigate how true this statement is, by evaluating the three heuristics used by IGRAPH against a variety of plans generated by different planners and heuristics.

5.18 Evaluating Potential Heuristic Bias

In the original complete goal-space model, both observer and the executing agent are assumed to be rational, and will never incorrectly estimate the distance to a goal. In the relaxed model this no longer holds, which leads to the seemingly obvious statement that both agent and observer may *not* be using the same heuristic.

While rather self-evident, this may have a profound impact upon recognition. If the observer is using a more accurate heuristic than the agent, then they can potentially produce better goal hypotheses, while if the opposite is true, the agent can reach the goal in a shorter number of steps. However, if both agent and observer are using the same *suboptimal* heuristic, then there is potential for the observer to have an advantage in determining the agent's goal (in a *keyhole* recognition context [1].). For instance, if the agent is in state S_n and the observer knows that because of the state's structure the agent will always choose action A_1, they can better predict the final goal being pursued. Indeed, if the heuristic is deterministic, it becomes possible to produce all possible plans the agent can possibly be pursuing — in turn reducing the recognition problem to that of classical recognition, wherein plans in a library are eliminated by observed evidence [97].

This section shall investigate whether this is a possibility in the domains tested. In particular, it will look at how IGRAPH performs on the same IPC3 domains that were considered in Section 5.17, when both the plan being observed and the heuristic used for observation are identical. Specifically, the FF heuristic as encoded in IGRAPH will be applied to plans generated with the original FF planner [88], the JavaFF planner [44], and LPG planner [66].

Note that here a distinction must be made between the FF heuristic, and what will be termed the JavaFF heuristic (and the respective planners). IGRAPH is built upon the code-base for JavaFF, which itself aims to be a Java implementation of the FF planner. However, in practice, JavaFF is not a perfect duplication, but for most problems will produce the same plan[19]. Therefore, for clarity, the FF heuristic has been used to generate plans with the FF planner, and an encoding of it has been implemented in IGRAPH (referred to as the JavaFF heuristic), and used again to generate a further set of plans which are referred to as JavaFF plans.

The reason for choosing the LPG planner as the counter to FF is that it is not a forward-chaining planner. Rather, it generates plans using *local search* through genetically-inspired means. Briefly, an initial plan is created quickly and naïvely which solves the goal but is probably of extremely poor quality (length). After this initial plan has been produced, the planner introduces local *flaws* into the plan, then attempts to find a solution for these which is shorter than the original plan. This partial-order planning approach allows the original plan to be optimised over time, which may lead to causal chains not encountered during the forward-search planning approach which FF uses.

The expectation of these results is that IGRAPH will show a performance improvement on those problems where the planner and recogniser share the same heuristic. This is exemplified by plans created with JavaFF, and configurations of IGRAPH which use the h_{ff}/JavaFF heuristic. Plans created with FF are also expected to show this behaviour, but as the JavaFF implementation is not an exact replica of the original, this will have a slightly poorer performance (but will still show a bias in the results).

Figure 5.35 shows the total F_1 accuracy for observations at 25%, 50%, 75% and 100% of plan observation. This is termed the total "relative" F_1 score and is used as a means of comparing the results of three different plan sources, where the total F_1 score is unsuitable for the same reasons given in Section 5.17. While, a shorter plan should theoretically have a higher score at these relative percentages of plan completion, having heuristic estimates which are the same across planner and recogniser may cause a similar pattern to emerge. Of the configurations shown in Figure 5.35, those using the h_{ff}/JavaFF heuristic are of the greatest interest, as these should inflate any bias towards plans generated using JavaFF and perhaps FF itself.

As is now to be expected, the general performance of each configuration echoes that of previous results where use of W_{SA} performs best, with the exception of

[19]Note that JavaFF as presented in [44] lacks many of the original features which made FF successful, such as helpful actions, goal-ordering and a deterministic heuristic. Almost all of these missing features have been added to the version used in testing. Only goal-ordering as specified in [101, 102] and the deterministic search function present in FF are not included.

Domain	FF	LPG	JavaFF
Depots	3	10	—
Driverlog	3	12	3
Rovers	12	—	6
Satellite	8	14	3
Zenotravel	12	10	2
Total	38	46	14

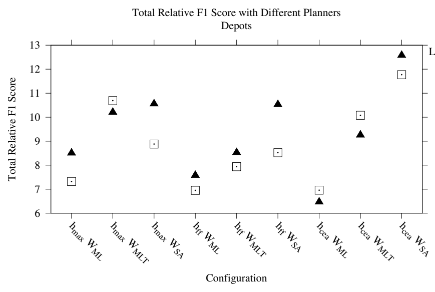
Table 5.11: The number of plans for each planner and domain, which are equal to the best plan-length used in testing. For example, in Rovers, the plan generated by FF has the shortest length in 12/15 tests, while JavaFF achieves this only 6/15 times. In this case, LPG has no results as it cannot produce an adequate set of solutions for all 15 problems.

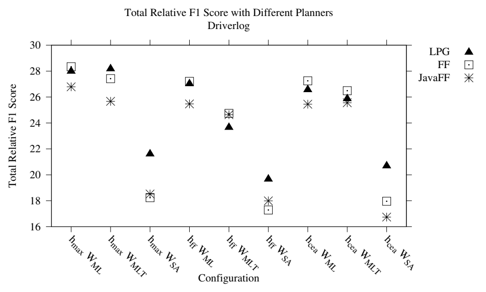
Driverlog where these configurations perform score approximately 25% lower than others. However, there is no overall bias towards configurations of IGRAPH which use h_{ff}, with the total score being similar across most domains no matter the configuration used.

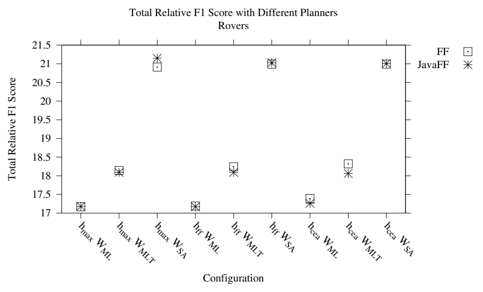
In most cases LPG has the highest scores on both Depots and Zenotravel, while on Satellite this is only achieved on configurations where W_{SA} is *not* used. Of the domains where JavaFF plans are available, there is no consistent behaviour between the use of this or the FF heuristic. For instance, FF generally slightly outperforms JavaFF on Driverlog, while the reverse is true on Satellite and Zenotravel. Scores for Rovers are almost identical.

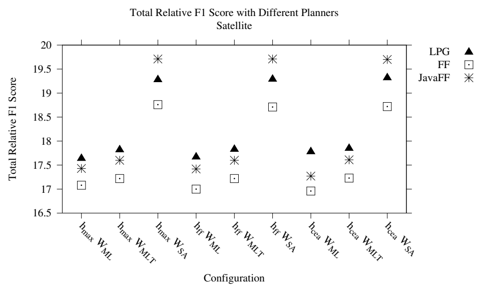
While these results appear to show no bias, it is not possible to state for certain that this is the case. However, it is conceivable that the source which has the shortest plans will receive the highest score as this should result in faster goal convergence. Table 5.11 shows the number of plans for which each planner finds the shortest solution. Overall, JavaFF matches the shortest plan only 14 times out of a possible 75 (five domains with 15 problems each), yet in recognition exceeds other plan-sources on some domains, such as Satellite where LPG is outperformed in configurations using W_{SA} and FF is dominated across all configurations. This is despite JavaFF only finding the shortest known solution for 3 problems, while FF and LPG find this 8 and 14 times respectively. It is therefore plausible that any bias in heuristic usage is domain-dependent.

The use of a heuristic (and indeed, a planner) to produce valid *plans* for each goal in $\mathcal{G}$ has previously been explored by Ramírez and Geffner [145]. Here,


(a) Depots


(b) Driverlog


(c) Rovers


(d) Satellite

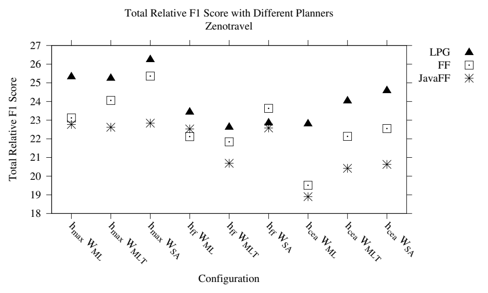

(e) Zenotravel

Figure 5.35: The total F_1 score for each configuration of IGRAPH when tested on plans generated using FF, JavaFF and LPG, and only hypotheses generated at 25%, 50%, 75% and 100% of observation are included. Note that there are no available results for Rovers using LPG or for Depots using JavaFF.

goals are single literals as in the relaxed goal-space presented [20], and a planner [76, 161] is used to produce plans which are in turn used as heuristic estimates. Such behaviour naturally produces both accurate goal estimates and would be more susceptible to bias — although it is unknown whether this was considered or indeed a possibility.

While the results appear to show that IGRAPH does not suffer from bias, having a model of the agent's heuristics with which to guide recognition is of course highly desirable. As stated previously, while the core argument of this thesis is that recognition should be possible without any prior knowledge, if such information is available or can be constructed at runtime, it should of course be applied. Learning models of agent behaviour is perhaps most akin to *policy learning* [153], and is closely related to classical plan recognition models where a plan-library can be used to construct a probabilistic policy — or vice versa [30, 38, 63].

5.19 On the Unexpected Performance of h_{max}

In the previous results, the performance of the h_{max} heuristic has often been surprisingly close to that of the supposedly more informed h_{ff} and h_{cea} heuristics. This is despite it offering extremely poor estimations for any goal which is more than a few steps from being achieved. This section enumerates some possible explanations for this behaviour.

5.19.1 Heuristic Admissibility

While h_{max} is admissible, both h_{ff} and h_{cea} are not, but do offer far more accurate estimates. However, at further goal distances these estimates become less reliable and, in certain domains, may increase when there should be a decrease (and this increase exceeds the true number of steps required).

This raises the question of whether heuristic accuracy is strongly linked to admissibility in goal recognition. In essence, an increase in the estimate to a goal indicates that the literal is not being pursued, while a stagnation of the estimates is inconclusive. It may be that h_{max} is achieving the same results as more accurate heuristics, by simply "ignoring" any heuristic movement until it is definitely closer.

[20]The model presented by Ramírez and Geffner does allow for conjunctive goals to be considered, but in practice only a small set of individual literals are enumerated and evaluated in their equivalent of $\mathcal{G}$.

5.19.2 Heuristic Estimate Similarity

The admissibility of h_{max} is also linked to a further possible reason for the overall performance. Ideally, when an agent pursues their goal, the estimate to the goal should lower while other mutually-exclusive goals have their estimates increase (or at least stay the same). However, in practice this behaviour can be unlikely, with multiple goals becoming closer at the same time — such as the possibility of communicating a rock sample instead of image analysis results in Rovers, or a passenger getting off at a given airport versus staying on the plane and disembarking at the next airport in Zenotravel.

One possibility is that regardless of heuristic choice, in most cases when $h(G^*)$ lowers, the estimates to most goals in the true goal's sub-goal-space *also* decrease (or also increase/stay the same). This would lead to similar posterior probabilities being generated after the observation, as it is not important how many times a goal has gotten closer, if all mutually-exclusive goals demonstrate the same behaviour regardless of heuristic.

As an example, consider the relaxed goal-space $\mathcal{G} = \{G_1, G_2, G_3\}$. At initialisation, G_1 is true and G_2 and G_3 both have an estimate of five. The agent is then observed executing four actions which lower this estimate to both unachieved goals, before finally executing a fifth action which achieves G_2 only. In this scenario where an optimal heuristic estimate is available, it is impossible to determine whether G_2 or G_3 is the true goal until the last observation. As the estimates lower at the same time, they will have the same posterior probability, making any hypothesis a tie-break between them.

Now consider that h_{ff} is used, and that it believes that after the second observation both G_2 and G_3 are further away from being achieved. This will not affect the output of IGRAPH — both goals have the same probability until the last observation. The same is true of h_{cea}, which may offer more accuracy but can ultimately produce inadmissible estimates.

However, if h_{max} believes that both unachieved goals are five steps away at the beginning of observation, then it is guaranteed that this value will not increase given the observed plan. Therefore, as long as at least a single observation lowers the distance to G_2 and G_3, there will be no difference in the hypotheses produced by h_{max}, h_{ff} or h_{cea} prior to the final observation (given that tie-breaking is deterministic).

While this example is perhaps contrived in order to demonstrate a possible

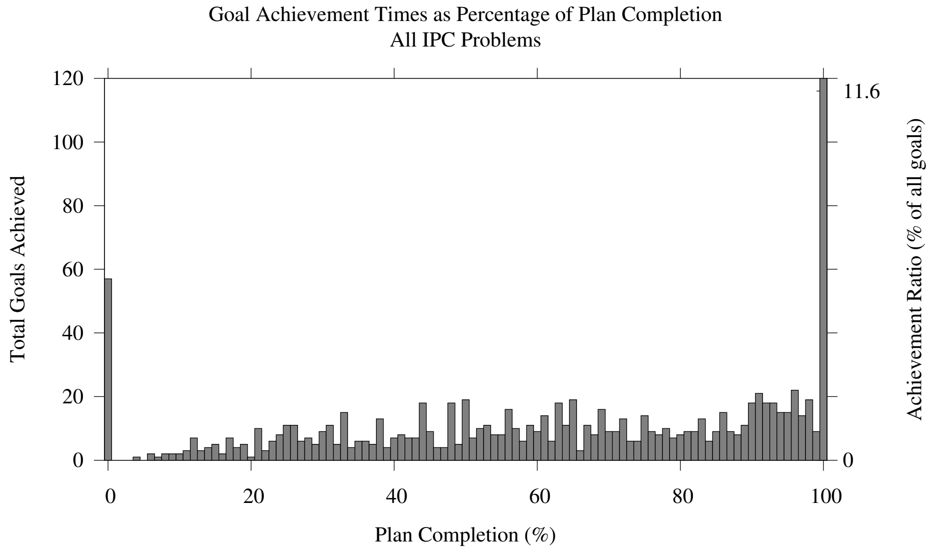


Figure 5.36: The distribution of goal-achievement times, normalised by overall plan length for all IPC problems, using the shortest solutions taken from IPC3/IPC5. Goals which are achieved on the final observation (100% completion) account for only 11.6% of the total goals achieved.

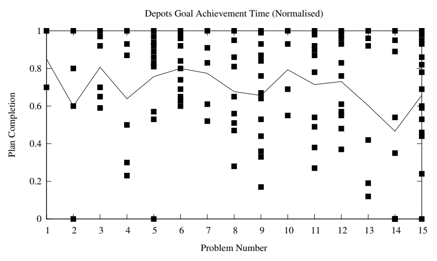
explanation for the performance of h_{max}, such behaviour is nonetheless possible during observation.

5.19.3 Achieving Goals Throughout Observation

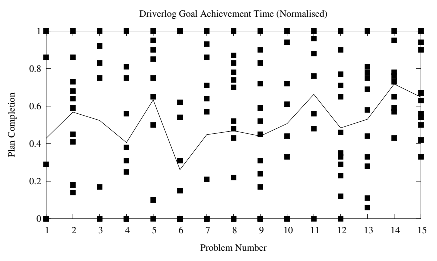
As has been covered previously, h_{max} is known to be poor at producing accurate estimates for facts which are more than a few steps from being achieved. However, while the final goal in the IPC domains tested is often conjunctive, the achievement time of the individual literals tends to be scattered throughout the plan.

As a visual aid, Figures 5.36 and 5.37 show the normalised times at which goal-literals are actually achieved during the plans used for evaluation. As these demonstrate, for non-trivial problems goals are often achieved almost uniformly *throughout* execution/observation.

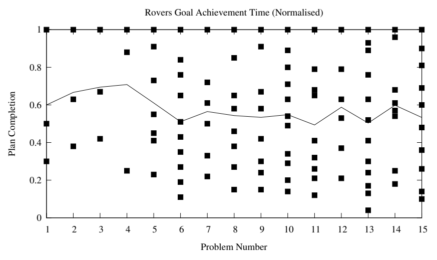
While it is unknown which goal the planner is pursuing at each step in the planning process (if any), these results appear to indicate that the difference in heuristic estimate *between* consecutively achieved goals is often a small value (relative to overall plan length). This may allow the performance of h_{max} to approach other more informative heuristics, given that the relaxed goal-space contains only individual literals. For instance, after the achievement of goal G_1,



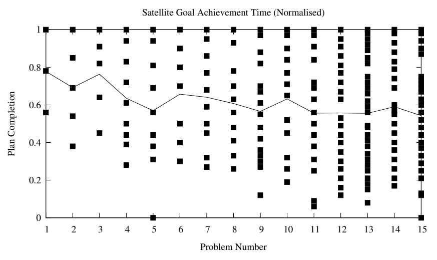
(a) Depots



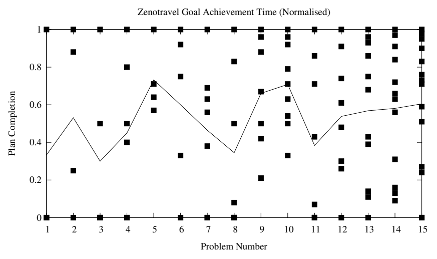
(b) Driverlog



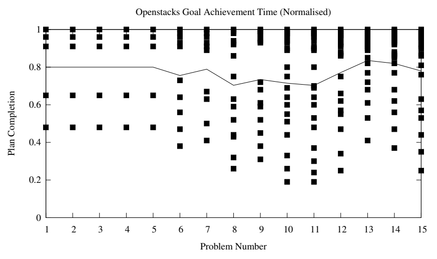
(c) Rovers



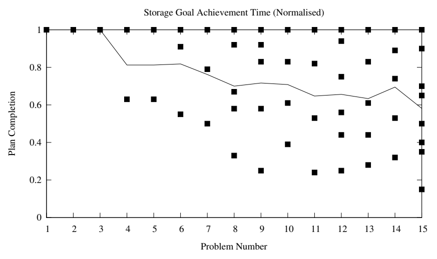
(d) Satellite



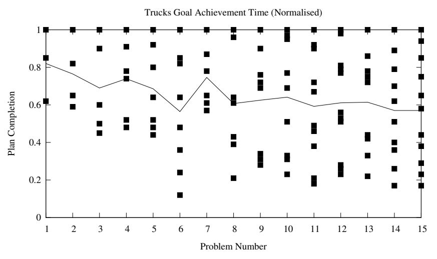
(e) Zenotravel



(f) Openstacks



(g) Storage



(h) Trucks

Figure 5.37: The individual goal achievement times as a ratio of per-problem plan completion. Each figure shows that for non-trivial problems, goals are achieved throughout plans rather than at the end (although the achievement of the final unachieved goal will always denote plan completion in a well-written STRIPS plan). The average goal-achievement time on each problem is plotted as a line.

the estimate from the achieving state to goal G_2 may be small enough such that $h_{max}(G_2) = h_{ff}(G_2) = h_{cea}(G_2)$, resulting in no substantial difference between heuristics.

Even if this is not the case, and the estimates are not equal, this behaviour will be approximated as the variance between each heuristics estimate reduces. This "lag" may be what causes the total F_1 score for h_{max} on intermediate hypotheses to be only slightly lower than those of h_{ff} and h_{cea}.

In reality, it is likely that the true reason for the uniform results is a mixture of the above suppositions. It is certainly the case that domains in which h_{max} performs well are those where it is difficult for h_{ff} and h_{cea} to compute accurate estimates due to combinatorial search. These domains (Depots, Driverlog, Storage), all contain a block-stacking subproblem which h_{ff} and h_{cea} struggle with, leading to estimates fluctuating wildly between observations. On the other hand, h_{max} has an upper bound which usually only lowers once the goal truly has gotten closer. While the other heuristics are indicating that a goal is getting further away, h_{max} more often than not believes that it has not moved at all, causing the associated probability to remain the same across the respective sub-goal-space.

5.20 Chapter Summary

This chapter has investigated the heuristic-based model of goal recognition put forward in Chapters 3 and 4.

The results given show that it is largely possible to perform accurate goal recognition using a domain-independent formalism. Section 5.10 demonstrated that domain-independent heuristics taken from the planning community can be successfully used in determining an agent's goal across multiple example domains. This was shown to be applicable in both an online context through generating *intermediate* hypotheses and an offline context with *final* hypotheses.

Three suboptimal heuristics have been evaluated — the max [27], Fast-Forward [88] and context-enhanced additive heuristics [80] — each with their own benefits and drawbacks, along with three methods for deriving a value for the Bayesian likelihood function. It has been shown that heuristics which provide more accurate estimates have higher accuracy, although admissibility may play an important role in this.

Section 5.12 showed that the model enables accurate online *prediction* of the true goal for the majority of test cases. This section also demonstrated that 80–90% of goals are successfully hypothesised at some point during recognition. This indicates that the model is largely backwards-compatible with previous work on planning-based goal recognition [89], with the added benefits of predictive capabilities.

Beyond classical plan recognition scenarios, Section 5.13 showed that while estimates for the true number of observations remaining is inconsistent in its accuracy, the *bounded* hypotheses produced can be highly accurate in determining short-term goals. This has implications for online scenarios where a subject's plans are to be intercepted or even assisted by a virtual agent.

Investigations into the automatic derivation of prior probabilities at runtime based upon domain knowledge, have demonstrated that it is possible to bootstrap the recognition process during early observations. However, these benefits are only present if the domain exhibits certain properties (*strictly terminal*, distant goals, acyclic plans), and are often rapidly subsumed by observed evidence.

Section 5.16 showed how the heuristic-based model can be applied to detect goal abandonment, which was exemplified in a simple grid based and various city-navigation problems where the agent abandoned their goal from 1 to 99 times. Detection of goal abandonment has previously only been considered as an additional problem on top of an existing plan recognition model [61], as opposed to the heuristic-based model has implicit support.

Section 5.17 investigated the possibility of plans generated with domain knowledge offering better results than those using a domain-independent planner. This was done by considering the "relative" accuracy of hypotheses at various points during observation and the resulting difference in accuracy between plan sources. These results appeared to show that use of a domain-dependent plan did not cause any improvement in observation beyond that expected from using a shorter plan. Following on from this, the issue of heuristic bias was explored in Section 5.18. Like the domain-dependent plan sources, there appeared to be no visible bias towards plans which made use of the same heuristic as the observer. Finally, Section 5.19 presented some possible explanations for the performance of the h_{max} heuristic, which exceeds the predictions given in Section 5.6 stating that less informed heuristics should have lower recognition performance than more informed heuristics.

Chapter 6

Conclusions

6.1 Overview

This thesis has shown that it is possible to perform goal recognition using a heuristic-based model, rather than the traditional use of plan-libraries. Chapter 2 introduced and motivates the need for this model, before a complete model of the problem is presented in Chapter 3. This was then relaxed in Chapter 4 to provide a tractable implementation of heuristic-based recognition. This relaxed model has been evaluated thoroughly in Chapter 5, wherein it was shown to be able to produce accurate hypotheses across a range of applications.

6.2 Summary of Contributions

The heuristic model presented has several advantages over previous recognition models [38, 70, 97], which address neglected areas of the recognition problem, or enable further application of the research. The original contributions of this work as set out in Section 1.4 are revisited here.

- **Reliance of Plan Libraries**
  By removing the need for a plan-library to be present goal recognition is less-constrained, leading to a more complete model of the underlying problem. Sections 3.2 and 4.3 described how automatic generation of the goal-space, along with automatic reachability analysis allows the behaviour of libraries to be approximated without the need for human interaction. Section 3.3 detailed how heuristics can be used as a means of determining which members of the goal-space are being converged upon after each

observation. This enables rapid deployment of the system on a variety of problems, which would not otherwise be possible without a plan-library or domain expert being available.

- **Suboptimal Agents and Goal Abandonment** The use of heuristics and a planning-based representation allows for suboptimal agents to be observed without generation of a plan-library. While this means that the agent's plan length will be unknown, Section 4.3.7 demonstrated how heuristics could again be used to estimate the number of steps remaining based on the most probable hypothesis. This bound, $1 \leq b \leq \tau_{max}$, was in turn used to generate multiple bounded hypotheses representing the agent's most probable goal in the next b timesteps.

  Virtually all library-based models assume that the agent will pursue the same goal throughout observation. This assumption is also used throughout the heuristic-based model, but unlike library-based approaches, goal abandonment is implicitly supported. This is in contrast with previous work where abandonment must be computed as a separate process on top of the recognition problem.

  Section 4.3.9 demonstrated how the heuristic-based model instead requires only observed evidence to detect the abandonment of a goal. This widens the application of the model to real-world scenarios where agents are both suboptimal, may change goals mid-plan, or even situations where the subject is attempting to hide their true goal.

- **Non-Uniform Initial Distributions**
  Part of the functionality of the plan-library is to provide a prior distribution over the goal/plan-space. This distribution can only be applied if a domain expert has generated it, or repeated observations of subjects are available. Section 4.4 showed that this can be replaced with basic analysis of the underlying problem, such that a non-uniform distribution across the goal-space can be generated which bootstraps early hypotheses.

- **Standardisation**
  Previous work in recognition has often been unique, with no cross-testing of domains/plan libraries. This work uses a standardised input language in the form of PDDL 2.1, enabling recognition to be performed on any valid domain.

Beyond these criteria, the heuristic model also has compatibility with several features of previously-accepted recognition models, particularly that of Goldman, Geib and Millar [70]. In particular, *negative evidence*, *pending sets* and *partially-ordered* plans are accounted for. The model also implicitly supports plan and goal abandonment [61], as well as goals which remain true throughout observation.

6.2.1 Interesting Observations

Chapter 5 evaluated the relaxed heuristic model, and raised several interesting observations relating to the recognition problem.

The first of these is that heuristic accuracy may not be as important as theoretical results would indicate. While configurations of IGRAPH which used the h_{max} heuristic had the lowest overall performance, the associated score was not so low as to indicate that the heuristic was of no use in recognition. Given that the computation time associated with this heuristic is trivial, there may be specific applications where it is of use, such as environments with limited processor-time (video games) or where the domain exhibits certain properties (a transport problem such as that shown in Figure 5.29).

While only partially explored in Chapter 5, the use of hypotheses as goal-state estimations rather than purely goals offers interesting possibilities in integrating recognition with a planner. For example, given a state-hypothesis, a planner acting in a co-operative manner may be able to plan to the hypothesis-state more quickly using the additional facts. Conversely, the planner may inform the hypothesis-construction process by informing the recogniser that the hypothesis is unreachable.

6.3 Future Work

IGRAPH can be seen as a first-step into the world of heuristic-based and library-free goal recognition. Now that the base-formalism has been defined and evaluated, there are many avenues of continued research which can be explored. This section will provide a brief overview of these.

- **Landmarks**
  In constructing a plan, an agent may not decide to 'aim' directly for the end goal. Instead, they may decide to plan to specific *landmarks* which occur on the path to the true goal. In the strictest form, these are facts/goals which *must* be achieved prior to another goal, while in a more relaxed form

they include a *set* of facts, of which one must be true before another fact can be achieved. For example, in order to achieve the "at-work" goal, the agent will consider the landmark goals "leave-house" and "get-train" first, knowing that these will ultimately achieve the end-goal.

The use of landmarks in planning has been known of since 2000 [137, 138], and forms the backbone of the winning planner in the last International Planning Competition (satisficing track) [161]. The planner, LAMA, automatically detects the ordered landmarks of the (known) goal at runtime using domain analysis, then plans to each landmark in turn.

Having the ability to detect which landmarks an agent is using to guide their search can have powerful implications for recognition. Knowing that there is a partial or total ordering of landmarks between goals could enable more accurate inference. For example, the knowledge that the agent in the above scenario must first use some form of transport to get to their destination could allow the observer to more accurately infer that they are going to work. With an optimal heuristic at the observer's disposal, this would offer no benefit, but as the evaluation chapter has shown, heuristics can struggle to offer high-quality estimations for distant goals. Knowing that landmark L_1 precedes goal G_1, could allow the observer to increase the probability of G_1 being the true goal, as L_1 has become closer, while the heuristic estimate of G_1 has remained the same due to its distance from the initial state.

- **Conjunctive Goals**

  Incorporation of conjunctive goals is a natural progression of the heuristic-based model. The primary benefit of having conjunctions present in the goal-space is that heuristic estimates are often more accurate than considering each goal separately, which should increase accuracy and reduce the need for additional relaxed model features (such as thread graphs and heuristic bonuses).

  Inclusion of conjunctive goals could be performed through extending sub-goal-spaces to include domain-generated literals representing the conjunctive literals. However, this process may be unsuited to domains where all mutex sets cannot be determined, as unreachable conjunctions could be inserted into the goal-space. However, approximation of valid conjunctive goals from plan traces is also an interesting area of extension.

- **Co-operative/Adversarial Recognition**
  The domains evaluated in Chapter 5 are done in the context of keyhole recognition. However, the results presented indicate that further work is possible in the areas of co-operative and adversarial recognition. In the former, agents are assisted by having the recognising agent perform actions on their behalf, which move the goal closer to being achieved. Conversely, in adversarial recognition the goal of the agent must be prevented by intercepting the executing plan. Integration of IGRAPH with a planner in order to achieve either of these tasks is a simple exercise, which when combined with the standardised input formalism of PDDL could enable a wide range of applications.

Appendix A

Scheduling The Observed Plan

A.1 Overview

The underlying model used in IGRAPH assumes that all observations will be totally-ordered, such that observation O_n will always be guaranteed to have occurred after O_{n-1}. At its most basic level this means that the plan being observed can be represented as a linear series of actions. However, this will be an inaccurate representation for most plans, as there is often an element of *concurrency* or *parallelism* present.

Concurrency in a plan indicates that n actions may execute at the same time t, in order to achieve a single goal. *Parallelism* is a weaker form of concurrency in which the plan contains several *sub-plans* which may or may not achieve the same goal[1].

Parallelism is of particular interest in goal recognition, as it is entirely possible that the subject may be undertaking several parallel sub-plans, in order that multiple goals can be achieved at plan completion. For instance, goals A and B may be achieved using every odd-numbered observation, while goal C is achieved using even-numbered observations.

In the complete goal-space model, parallelism is of no use because all goals exist within $\mathbb{G}$ and an optimal heuristic is available. Thus all steps of the plan are considered useful for the true goal. However, in the relaxed goal-space model

[1]The term "sub-plan" is used in a different context to that of classical recognition literature, where it is used to describe a decomposition of a high-level action-task. Here, a sub-plan is a series of causally-linked actions which achieve a subset of the overall goal conjunction or an intermediate goal.

t	s	*Obs*
1	0	`load package1 truck1 loc0`
2	0	`load package2 truck1 loc0`
3	1	`drive truck1 loc0 loc1`
4	0	`load package3 truck2 loc0`
5	2	`drive truck1 loc1 loc2`
6	1	`drive truck2 loc0 loc4`
7	3	`drive truck1 loc2 loc3`
8	4	`unload package1 truck1 loc3`
9	2	`drive truck2 loc4 loc5`
10	5	`unload package2 truck1 loc3`
11	3	`drive truck2 loc5 loc6`
12	4	`unload package3 truck2 loc6`

Table A.1: The timestamps for a series of observed actions in both an totally-ordered and scheduled context. Column t indicates the time at which the action was observed, while column s indicates the earliest time at which the action could have been observed. The scheduled time can then be used in construction of the plan-thread graph.

presented in Chapter 4, modelling parallelism can lead to more accurate estimates for work expended on each goal, $W(G)$.

A.2 Plan Scheduling

In order for a *plan-thread graph*[2] as described in Section 4.3.6 to be constructed, each observation must be assigned a timestamp indicating its earliest possible application in the plan. This allows for partially-ordered plan steps within a totally-ordered plan to be accommodated, and is achieved by *scheduling* the currently-observed plan. Note that a timestamp is not an indication of when the action was observed, but rather when it *could* have been observed. That is, an action O_t can be observed at time t, but could have been observed as early as time s. For clarity, a scheduled observation is denoted O_t^s.

Algorithm 13 outlines the algorithm for scheduling a STRIPS-style plan [49]. Here, actions are assumed to have a fixed-length of 1, while timestamps $t \in \mathbb{Z}^+$, and $s \in \mathbb{N}^+$. A scheduled timestamp s is essentially an integer, as this is a STRIPS-based plan, however, small *epsilon values* are added to the scheduled

[2]While the term "plan-graph" may seem appropriate here, it is avoided as it has a prior and different meaning within the planning literature [23].

timestamp to preserve the order in which actions were observed. For example, observation $O_{t=1}$ and $O_{t=2}$ which are both applicable at the same timestamp, will have a final scheduled timestamp of $O_{t=1}^{s=1.0001}$ and $O_{t=2}^{s=1.0002}$. The left-hand-side of the decimal place is referred to as the *major* timestamp, with the epsilon-value, κ, called the *minor* timestamp. Observations scheduled for later in the plan will have their κ value restart at 0.0001 (i.e. $O_{t=3}^{s=2.0001}$). This distinction between actions scheduled at the same major timestamp is a technical one which will be used during scheduling.

Regarding the input to the algorithm, I is the initial state and P_U is the unscheduled (totally-ordered) plan. In addition to these, a set of known mutexes M are passed to the algorithm. This mutex set must be *complete*, and therefore cannot simply be the set of binary-mutexes detected at runtime. The complete[3] set of Graphplan style action and fact mutexes [23] can be iteratively constructed throughout the threading process. These are as follows.

- Interference – Two actions are mutex if one deletes a precondition or add effect of the other.
- Competing Needs – Actions a and b are mutex if a precondition of a is mutex with a precondition of b. Two facts are themselves mutex if all actions which achieve them are mutex.

In addition to these mutex relations, a further relation is made explicit. While this is in fact a specialisation of the *interference* mutex, it is useful to fully specify it as it can often be overlooked.

- Blocking – If action a has a fact $f \in \{a_{add} \cup a_{del}\}$, then it is mutex with any other action b for which $f \in \{b_{add} \cap b_{del}\}$. By both deleting and adding the fact at the same time[4], b is said to be blocking achievement of f (where "achievement" can mean addition *or* deletion).

[3]The term "complete" is used in the context of the actions within the unscheduled plan. That is, the complete set of mutexes required by the scheduler need only include those members of the observed plan.

[4]The accepted principle within the planning community is that delete effects should always be applied prior to add effects.

Algorithm 13 schedule(I, P_U, M)

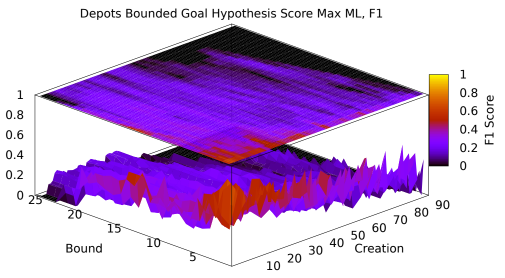
```
1: $P_S := \{\}$ {The scheduled plan}
2: $A_0 := \langle\{\}, I, \{\}\rangle$ {Stub action which adds initial state}
3: $lastAchievers := \{\}$ {Map of each facts last achieving action}
4: $requiredBy := \{\}$ {Map of facts required by scheduled actions}
5: $scheduled := \{\}$ {Map of timestamps to scheduled actions}
6: for all $F \in I$ do
7:     $lastAchievers(F) \leftarrow A_0$
8: end for
9: for all $A \in P_U$ do
10:     $latest := 0$ {The last action to achieve $F_{pc}$}
11:     for all $F_{pc} \in A_{pre}$ do
12:         $achiever := lastAchievers(F_{pc})$
13:         if $achiever_{major} > latest$ then
14:             $latest := achiever_{major}$
15:         end if
16:     end for
17:     $currentTime := latest$
18:     $mutex :=$ true
19:     while $mutex$ do
20:         $mutex :=$ true
21:         {Get actions already scheduled at $currentTime$}
22:         $existing := scheduled(currentTime)$
23:         if $areMutex(A, existing, M)$ then
24:             $currentTime := currentTime + 1$
25:             continue {Advance a single timestep and re-check}
26:         end if
27:         {Check that action does not delete any future scheduled actions preconditions}
28:         for all $F_{del} \in A_{del}$ do
29:             $requires := requiredBy(F_{del})$
30:             for all $r \in requires$ do
31:                 if $r_{major} \geq latest$ then
32:                     $currentTime := currentTime + 1$
33:                     goto 19 {Advance a single timestep and re-check}
34:                 end if
35:             end for
36:         end for
37:         {If this point has been reached, the value of $currentTime$ is a valid schedule time}
38:         $mutex :=$ false
39:     end while
40:     $scheduled(currentTime) \leftarrow A$
41:     for all $F_{pc} \in A_{pre}$ do
42:         $requiredBy(F_{pc}) \leftarrow A$
43:     end for
44:     for all $F \in A_{eff}$ do
45:         $achievedBy(F) \leftarrow A$
46:     end for
47:     $P_S \leftarrow A$
48: end for
49: return $P_S$
```

Appendix B

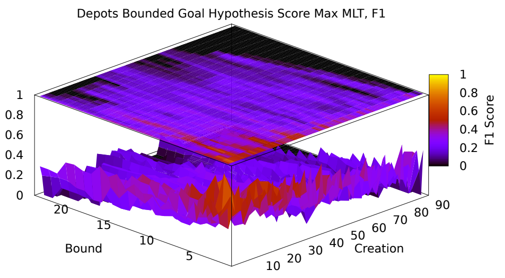
Bounded Hypothesis Results

This appendix contains visualisations of the results produced by *bounded hypotheses* as evaluated in Section 5.14. The figures presented here are averages of bounded hypotheses F_1, *precision* and *recall* scores over each domain and configuration tested. Each figure is a single configuration and domain pairing, with the score at a creation and bound time representing the average of all bounded hypotheses generated on the particular domain, across all problem files. That is, a score of $Hyp_4^{10} = 0.5$ means that the average score across all problems in the domain, at creation time $c = 4$ and bound time $b = 10$ is 0.5.

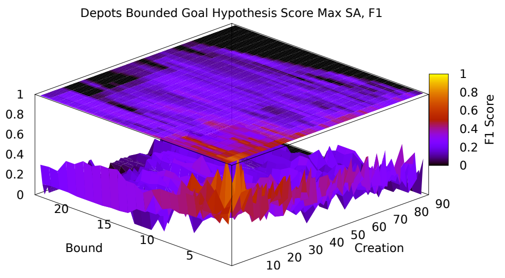
B.1 F_1 Score



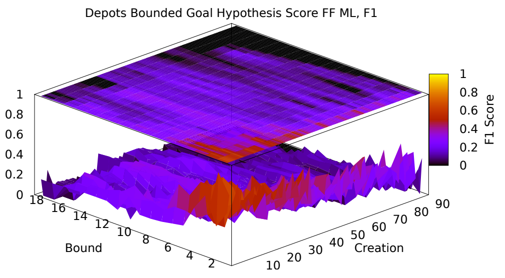
(a)



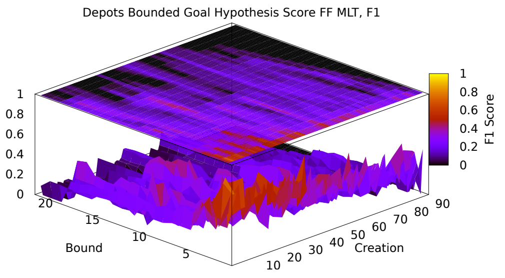
(b)



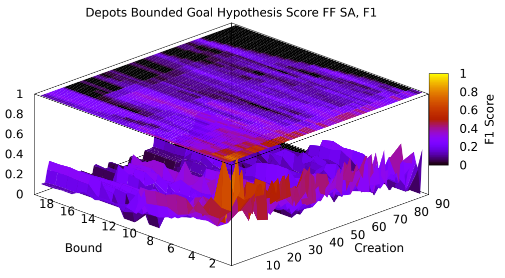
(c)



(d)

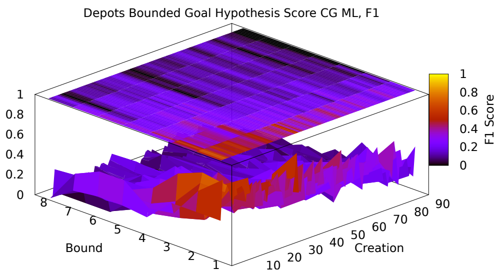


(e)

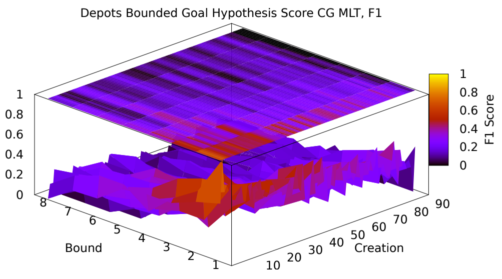


(f)

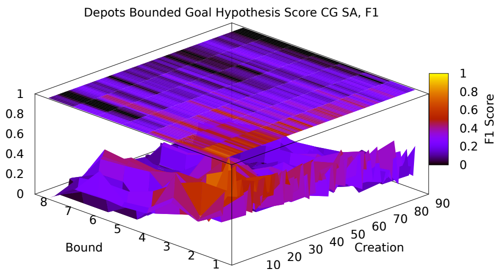
Figure B.1: F_1 bounded hypothesis results.


Depots Bounded Goal Hypothesis Score CG ML, F1
F1 Score
Bound
Creation

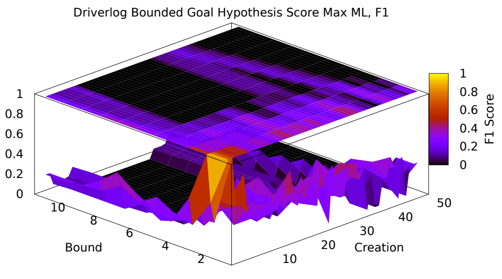
(a)


Depots Bounded Goal Hypothesis Score CG MLT, F1
F1 Score
Bound
Creation

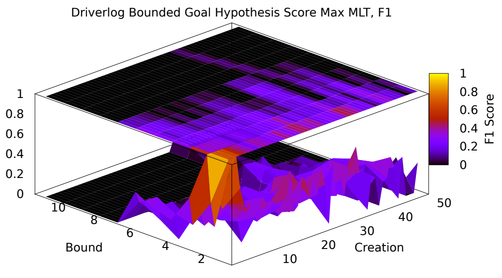
(b)


Depots Bounded Goal Hypothesis Score CG SA, F1
F1 Score
Bound
Creation

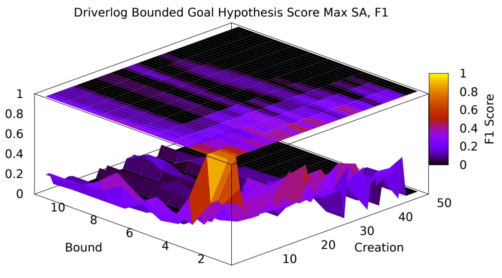
(c)


Driverlog Bounded Goal Hypothesis Score Max ML, F1
F1 Score
Bound
Creation

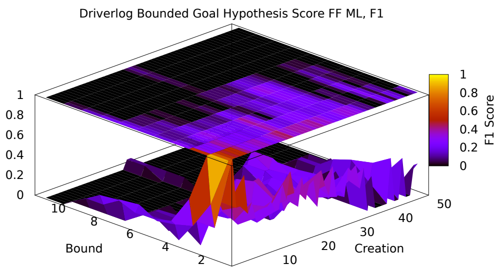
(d)


Driverlog Bounded Goal Hypothesis Score Max MLT, F1
F1 Score
Bound
Creation

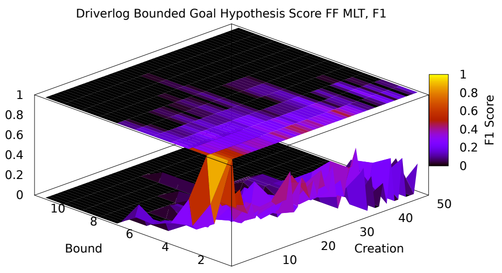
(e)


Driverlog Bounded Goal Hypothesis Score Max SA, F1
F1 Score
Bound
Creation

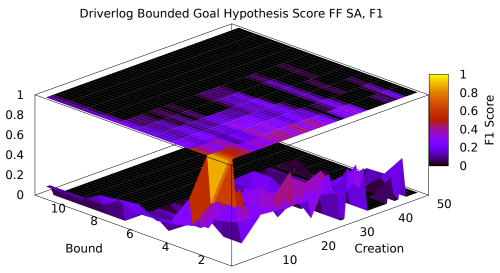
(f)



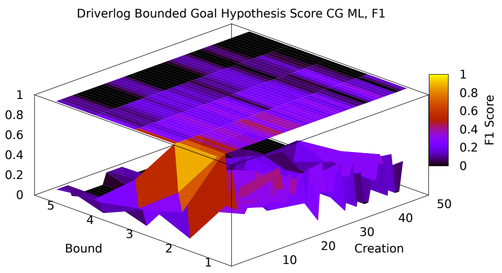
(a)



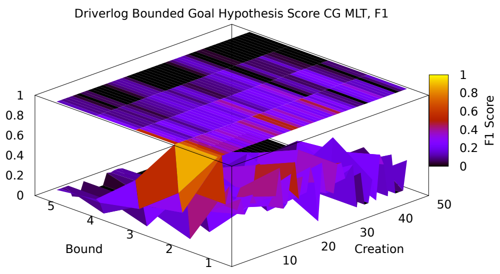
(b)



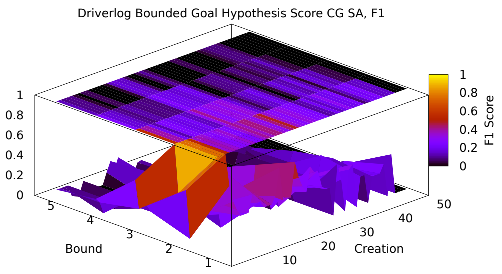
(c)



(d)

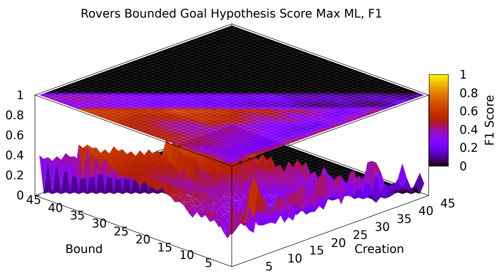


(e)

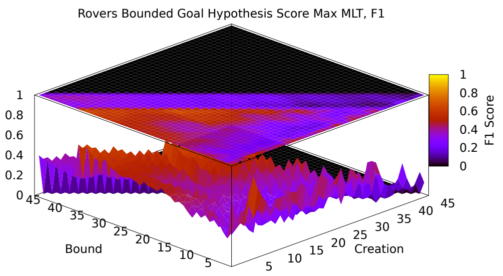


(f)

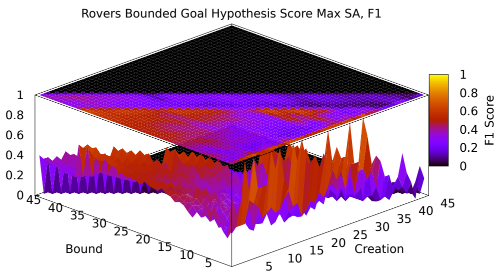
F_1 bounded hypothesis results.



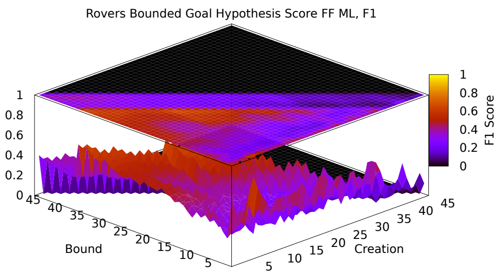
(a)



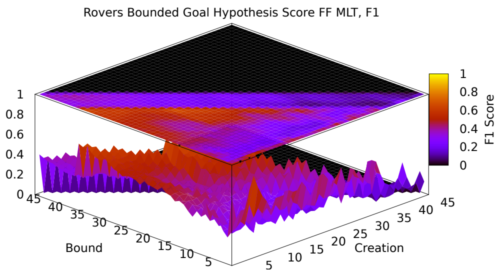
(b)



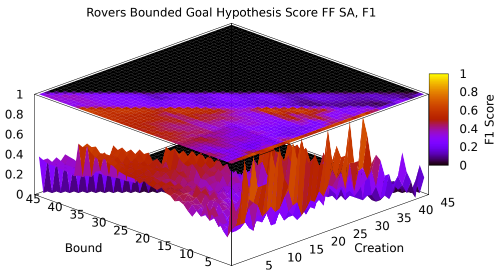
(c)



(d)

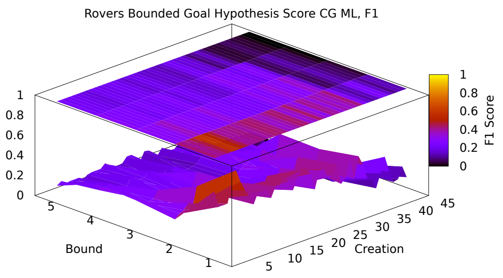


(e)

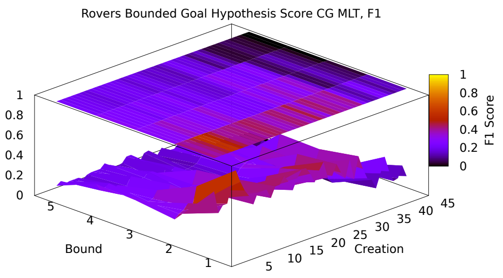


(f)

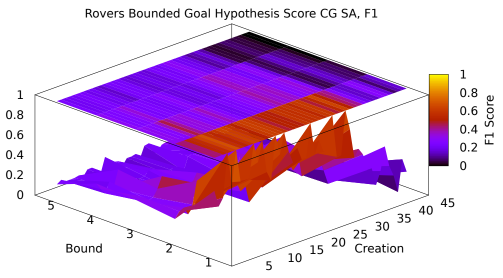
F_1 bounded hypothesis results.



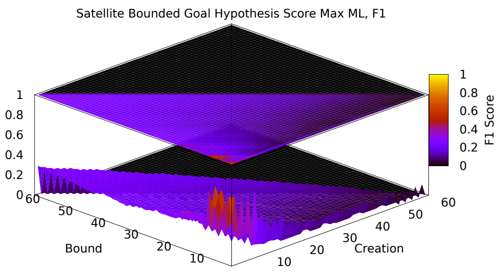
(a)



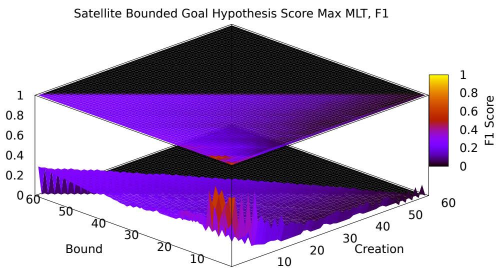
(b)



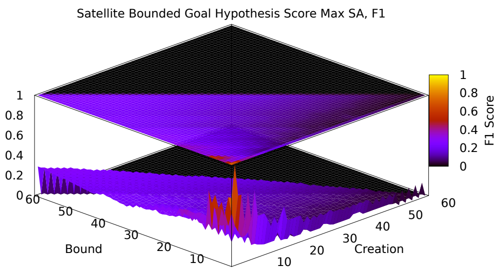
(c)



(d)

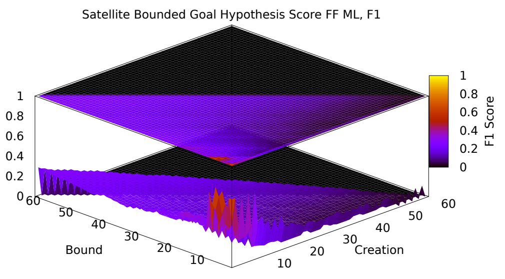


(e)

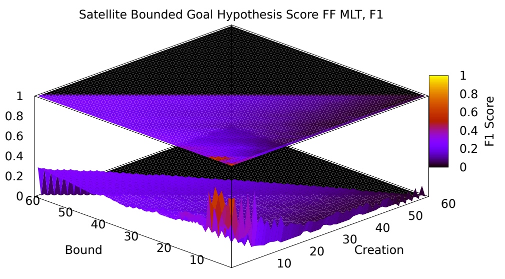


(f)

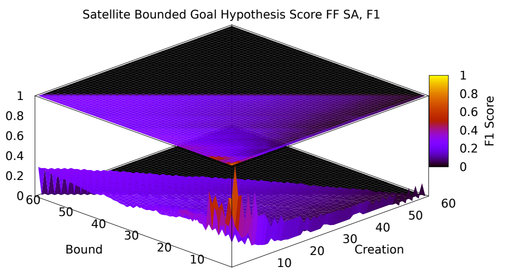
F_1 bounded hypothesis results.



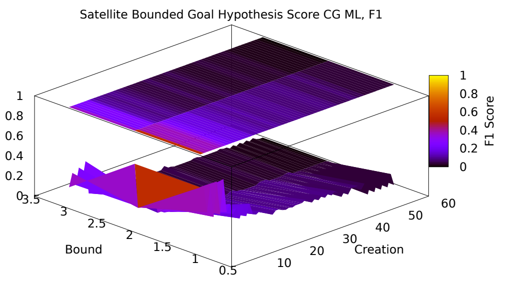
(a)



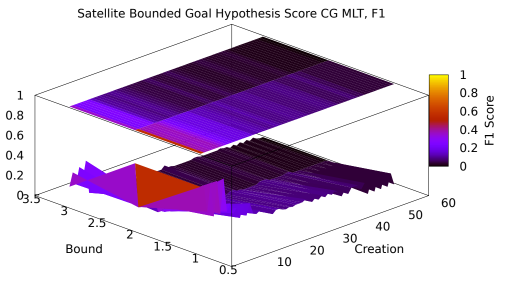
(b)



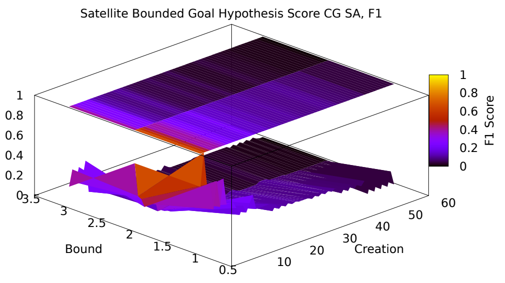
(c)



(d)

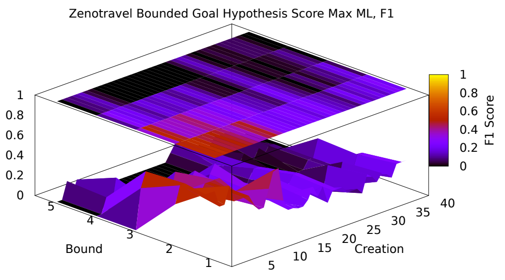


(e)

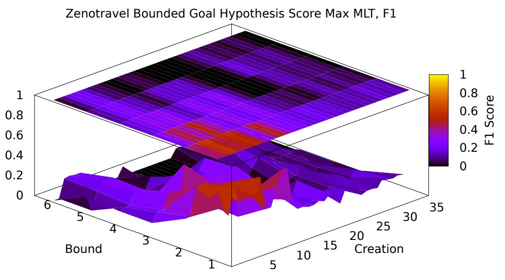


(f)

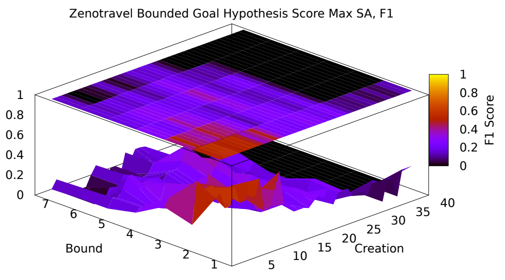
F_1 bounded hypothesis results.



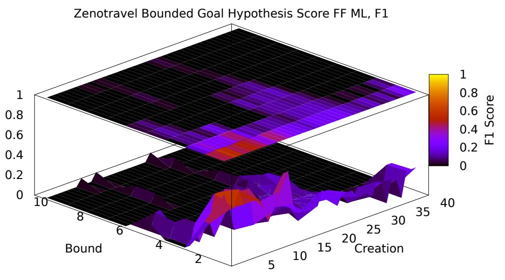
(a)



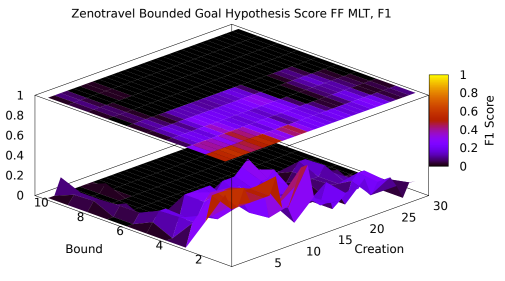
(b)



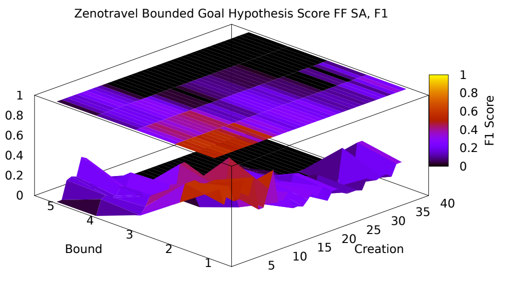
(c)



(d)

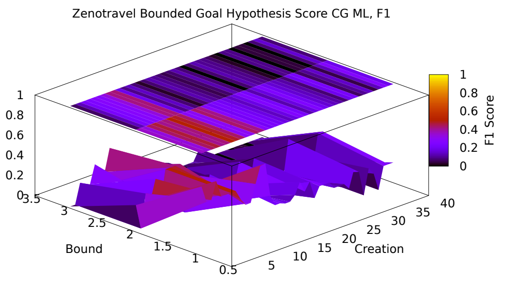


(e)

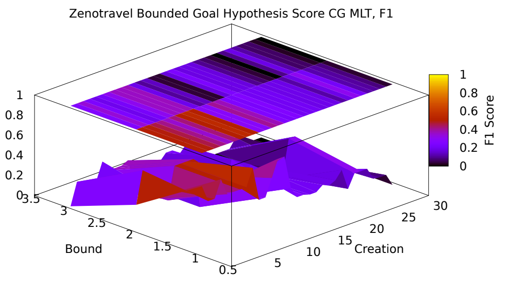


(f)

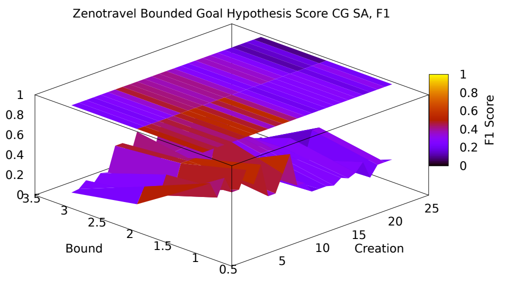
F_1 bounded hypothesis results.



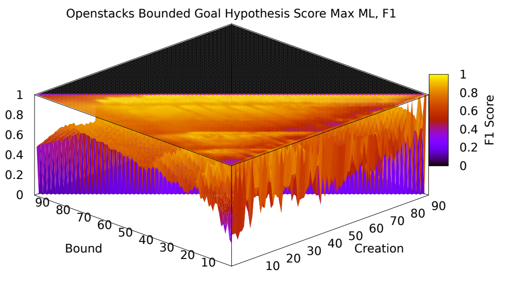
(a)



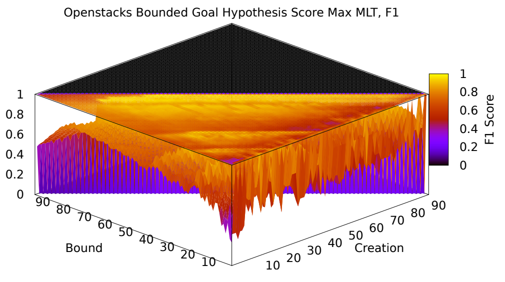
(b)



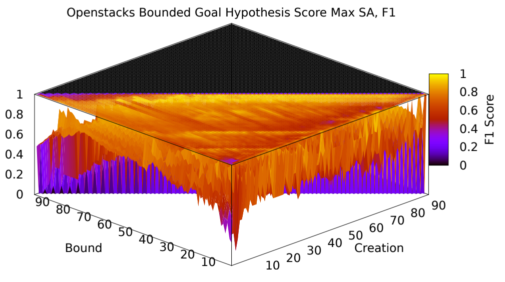
(c)



(d)

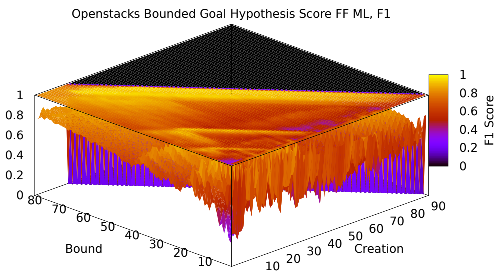


(e)

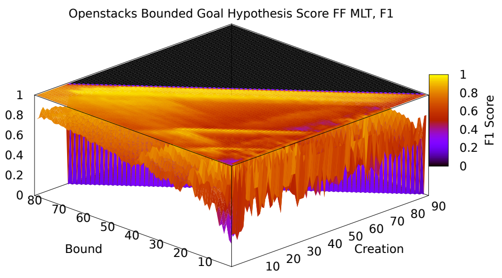


(f)

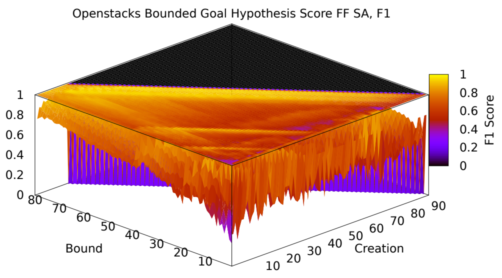
F_1 bounded hypothesis results.



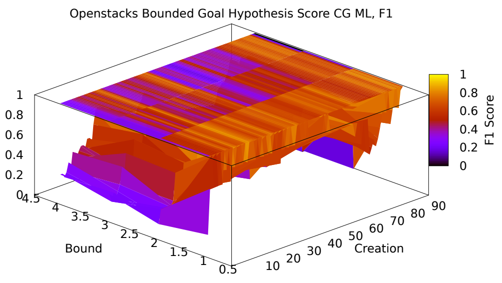
(a)



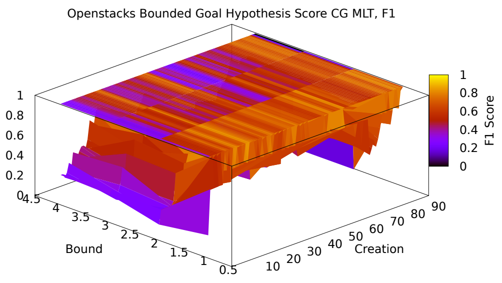
(b)



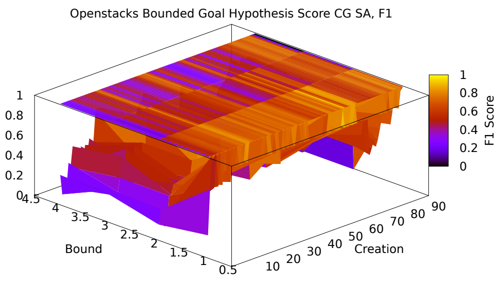
(c)



(d)

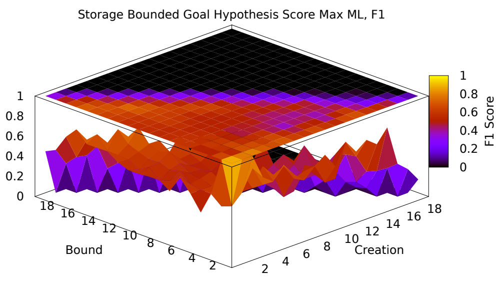


(e)

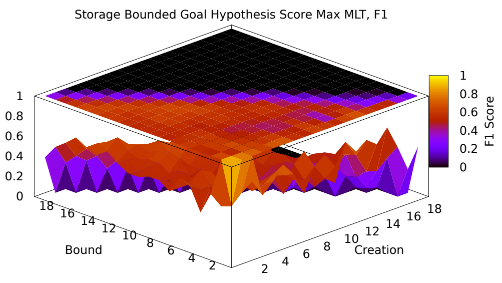


(f)

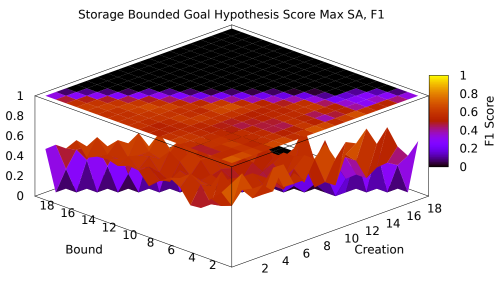
F_1 bounded hypothesis results.



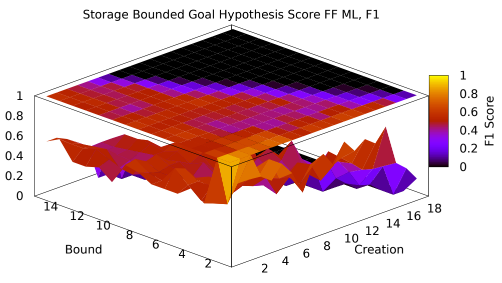
(a)



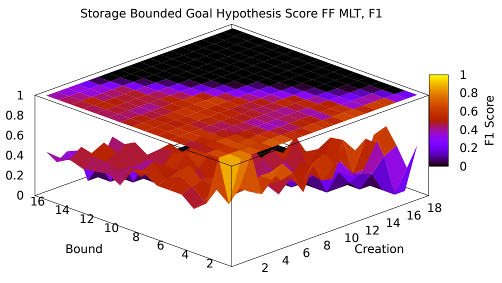
(b)



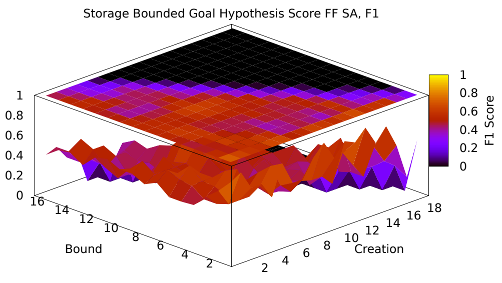
(c)



(d)

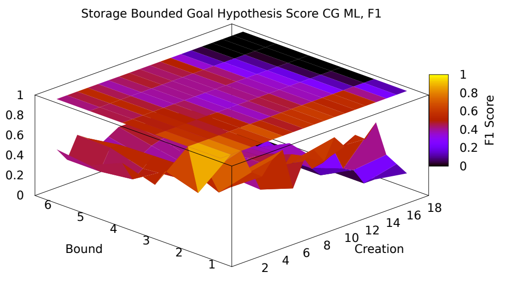


(e)

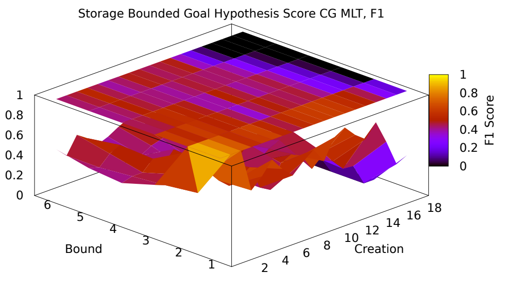


(f)

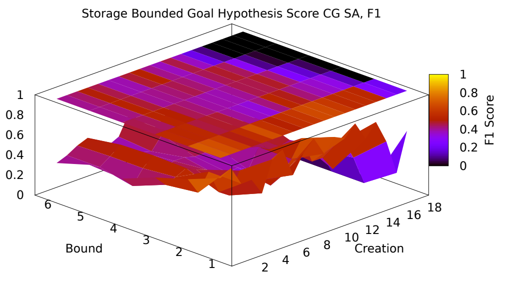
F_1 bounded hypothesis results.



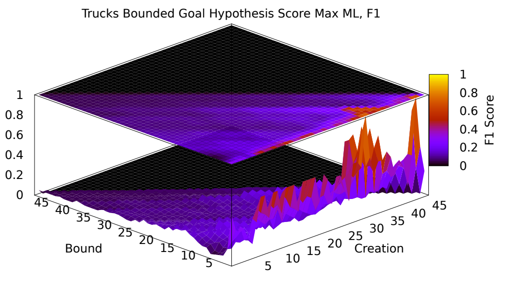
(a)



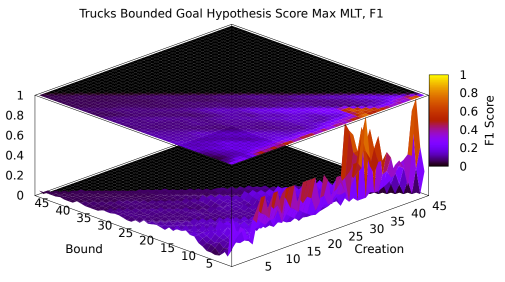
(b)



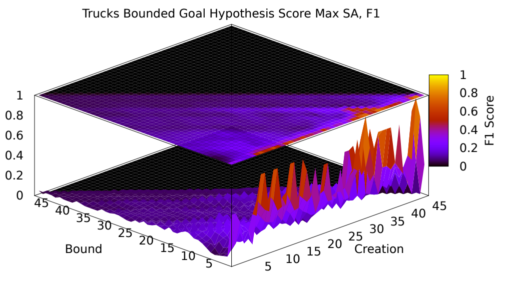
(c)



(d)

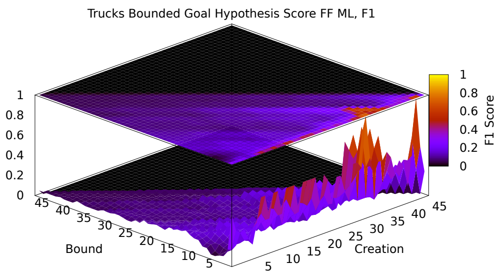


(e)

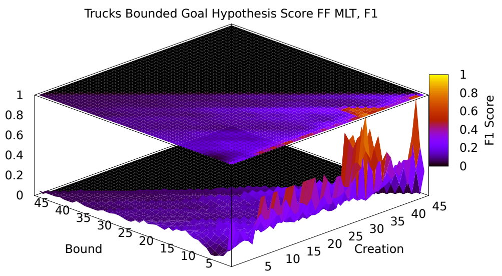


(f)

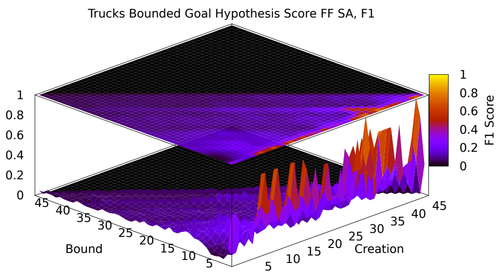
F_1 bounded hypothesis results.



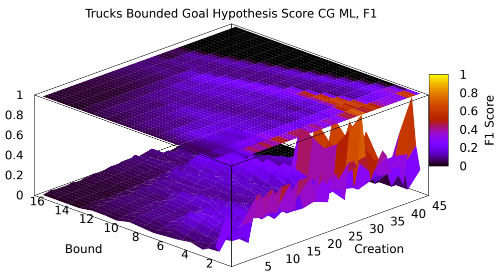
(a)



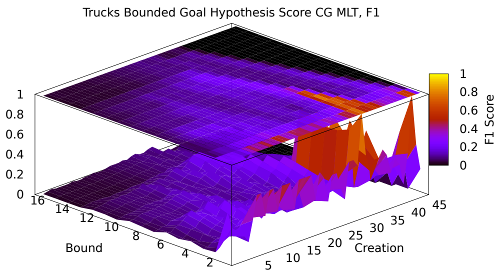
(b)



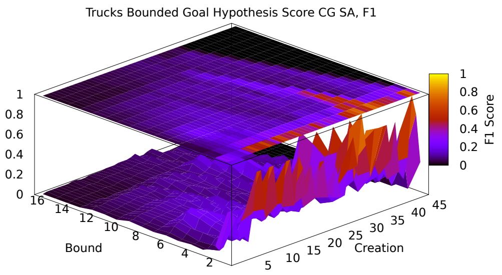
(c)



(d)



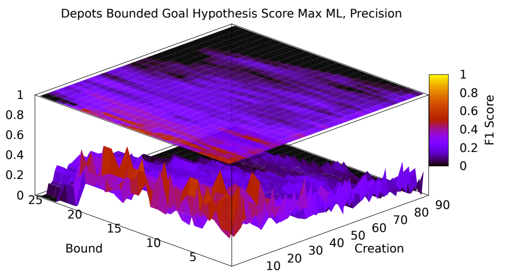
(e)



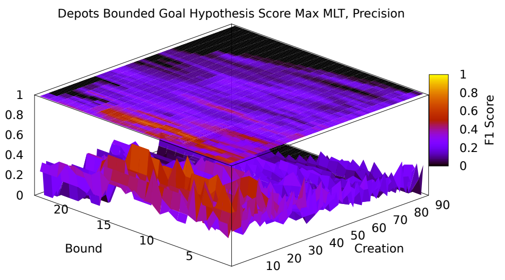
(f)

F_1 bounded hypothesis results.

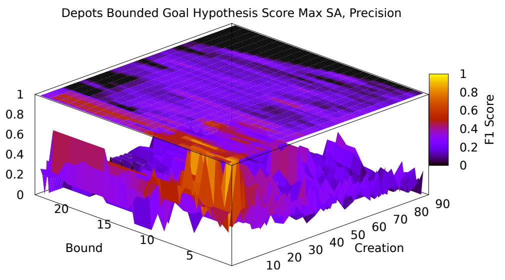
B.2 Precision



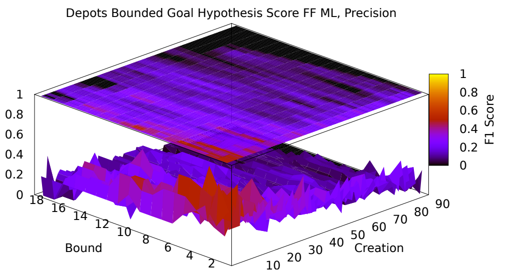
(a)



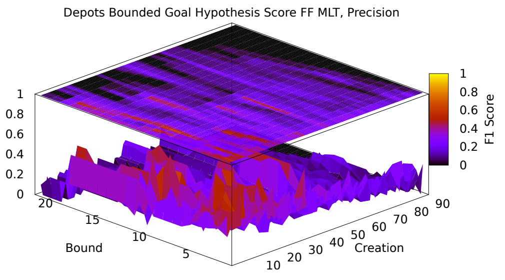
(b)



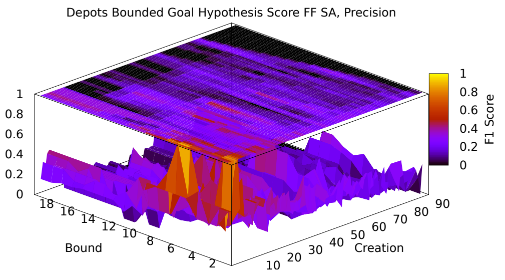
(c)



(d)

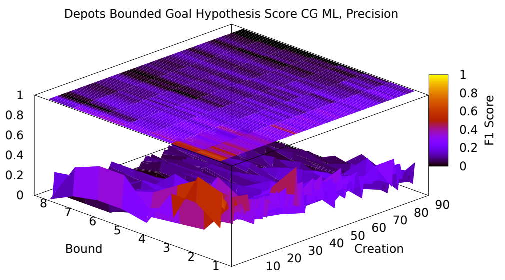


(e)

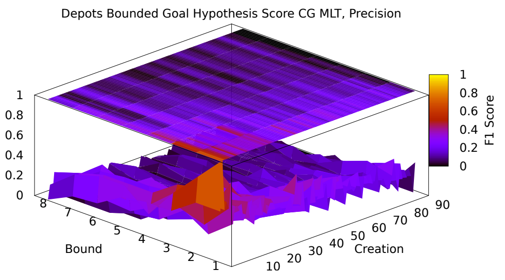


(f)

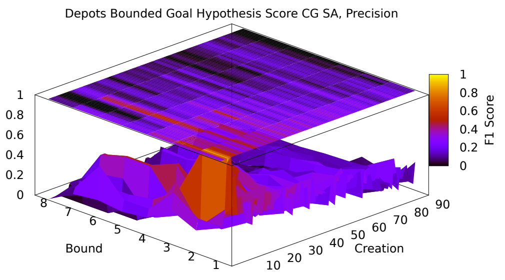
Figure B.2: *Precision* bounded hypothesis results.



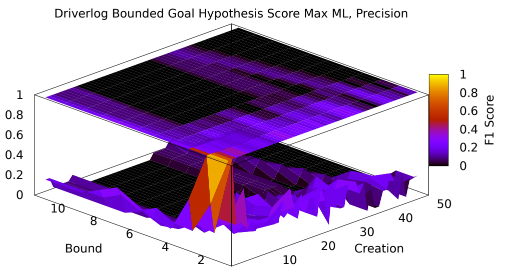
(a)



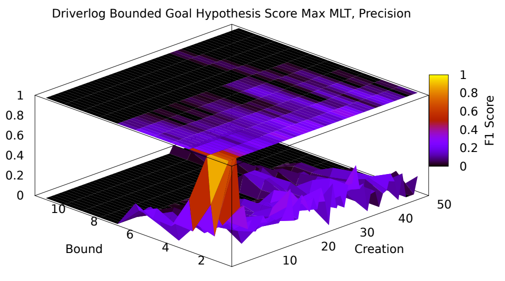
(b)



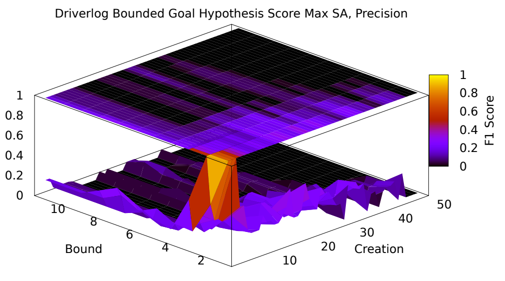
(c)



(d)

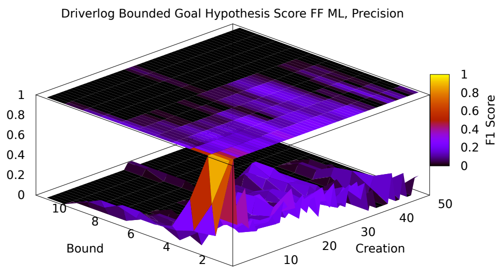


(e)

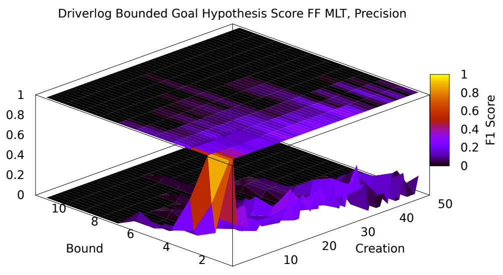


(f)

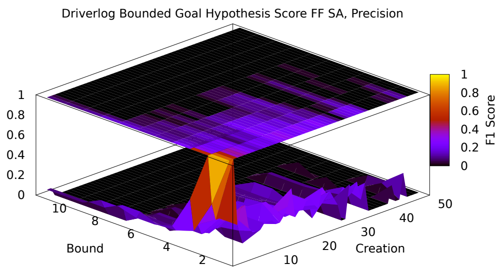
Precision bounded hypothesis results.



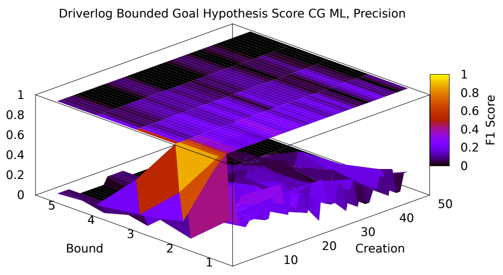
(a)



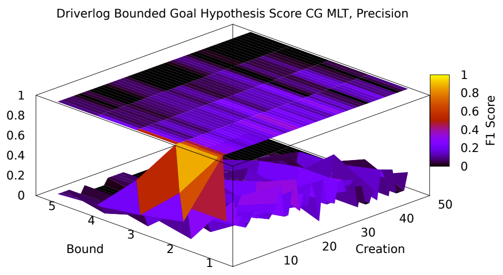
(b)



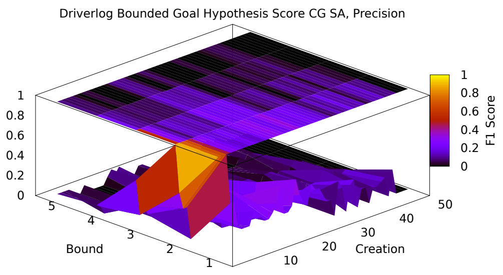
(c)



(d)

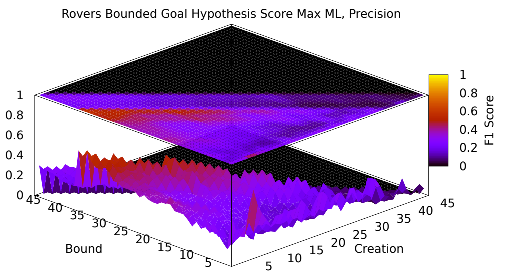


(e)

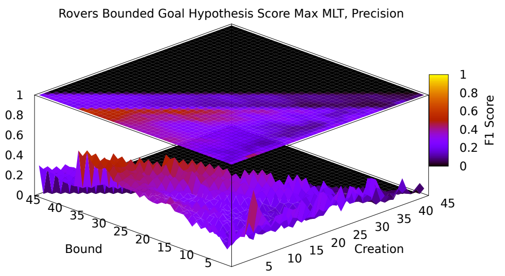


(f)

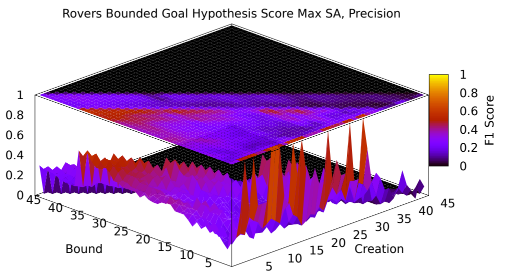
Precision bounded hypothesis results.



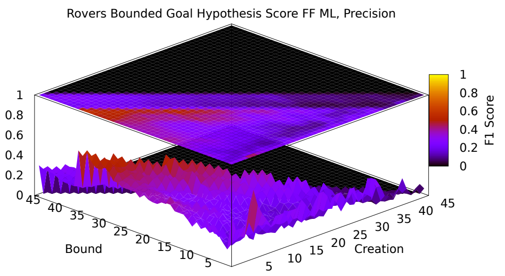
(a)



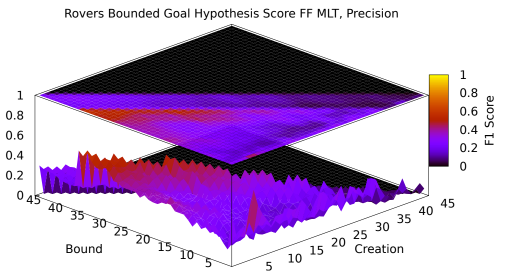
(b)



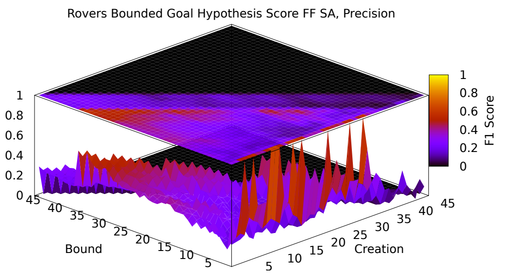
(c)



(d)

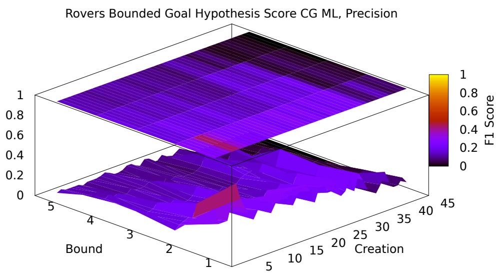


(e)

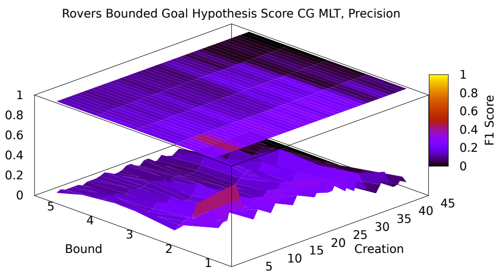


(f)

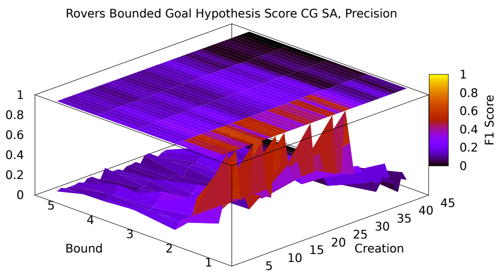
Precision bounded hypothesis results.



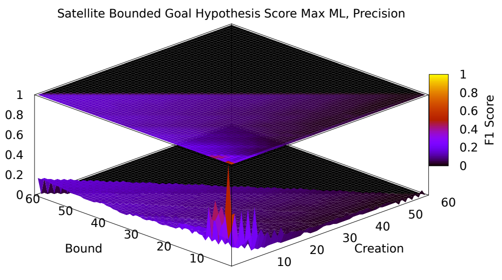
(a)



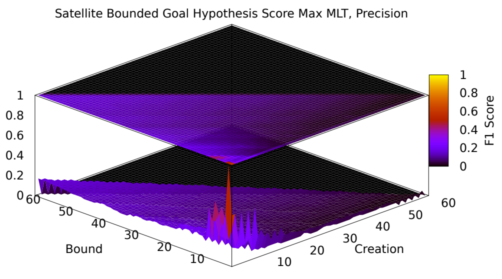
(b)



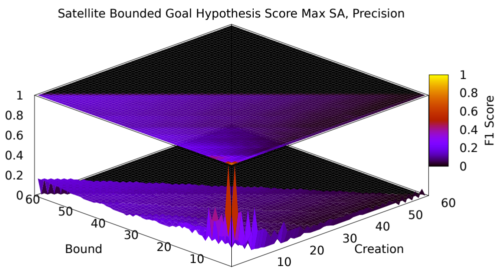
(c)



(d)

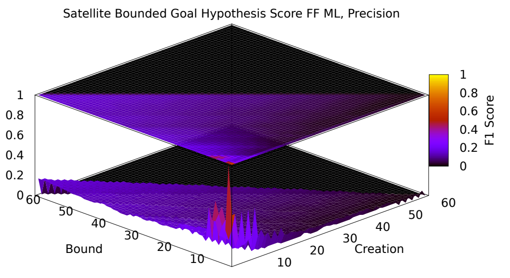


(e)

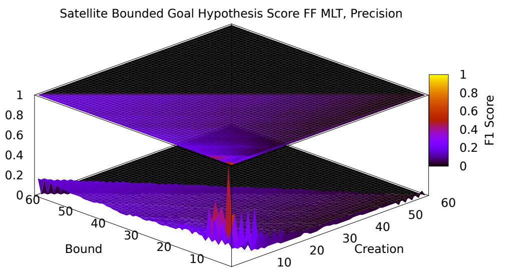


(f)

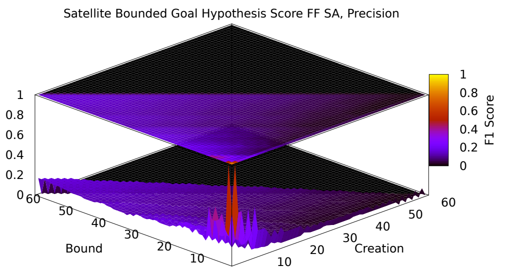
Precision bounded hypothesis results.



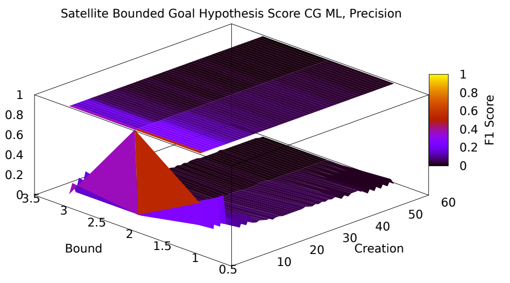
(a)



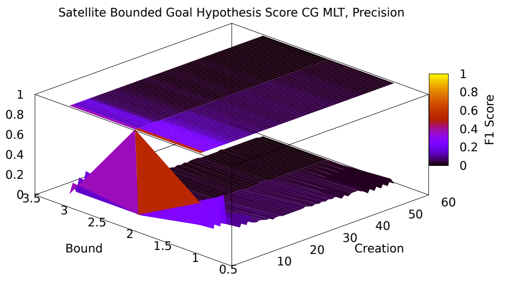
(b)



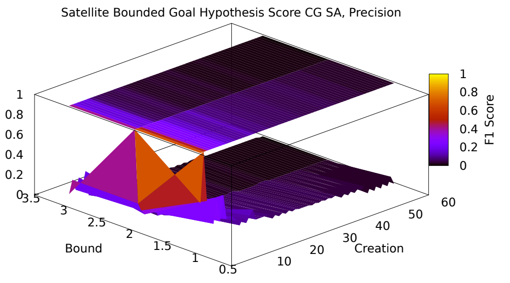
(c)



(d)

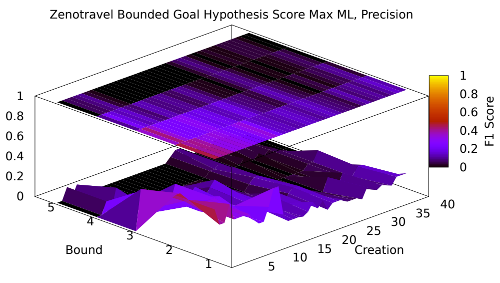


(e)

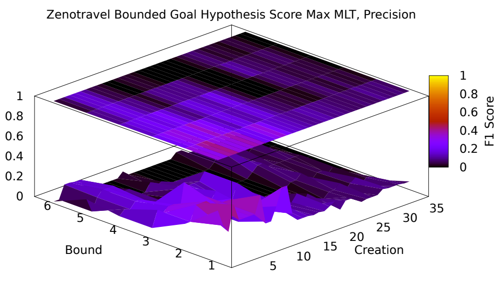


(f)

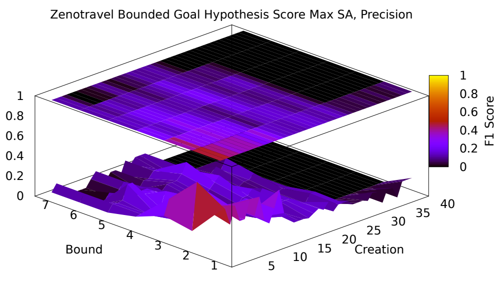
Precision bounded hypothesis results.



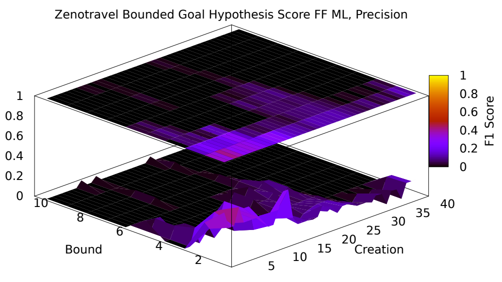
(a)



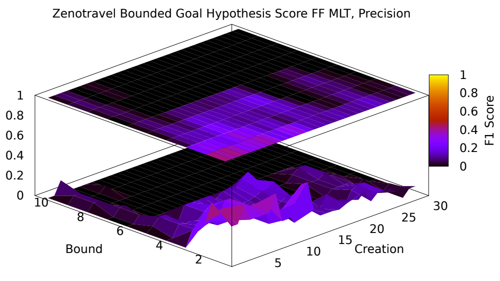
(b)



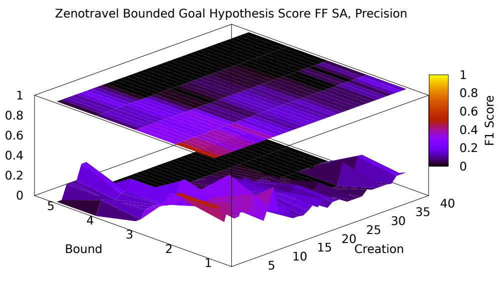
(c)



(d)

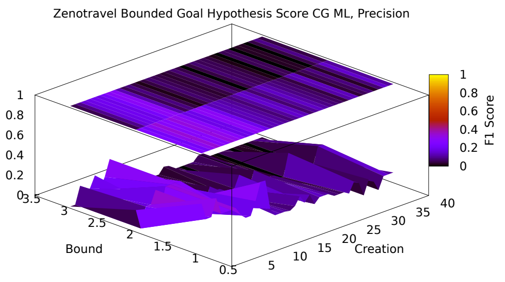


(e)

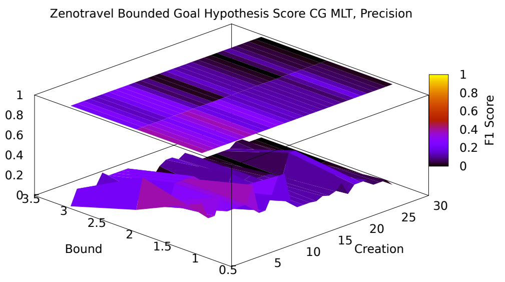


(f)

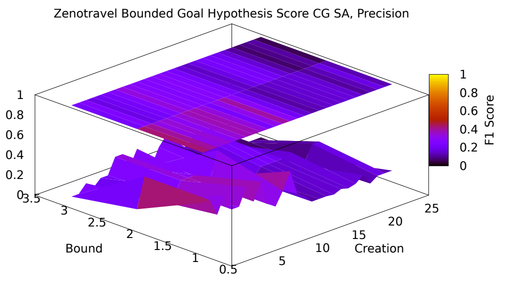
Precision bounded hypothesis results.



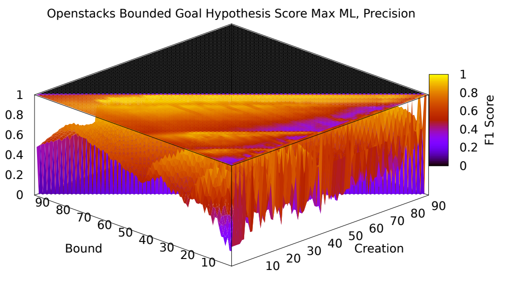
(a)



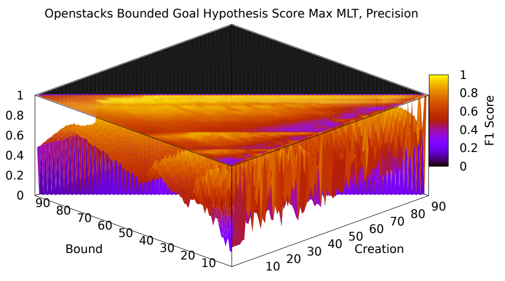
(b)



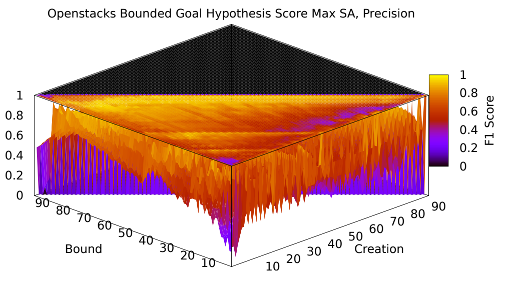
(c)



(d)

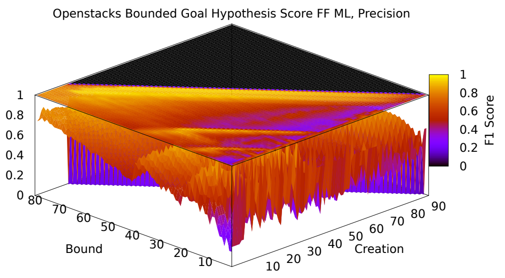


(e)

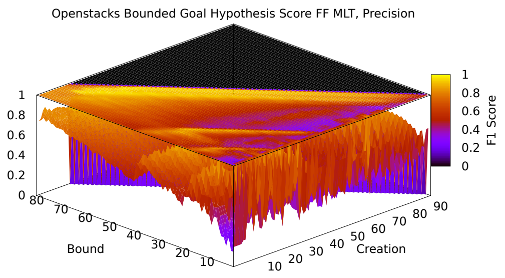


(f)

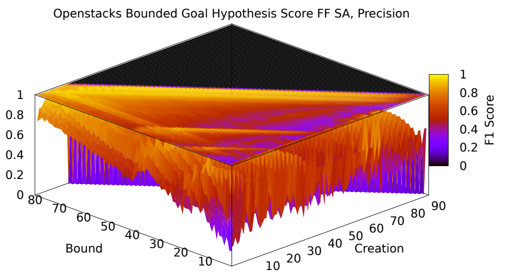
Precision bounded hypothesis results.



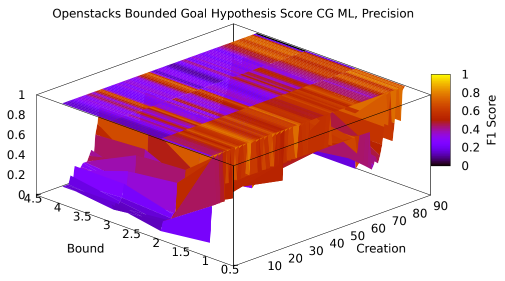
(a)



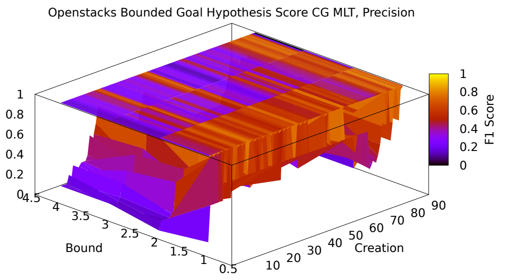
(b)



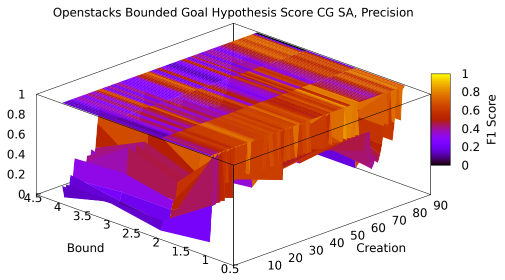
(c)



(d)

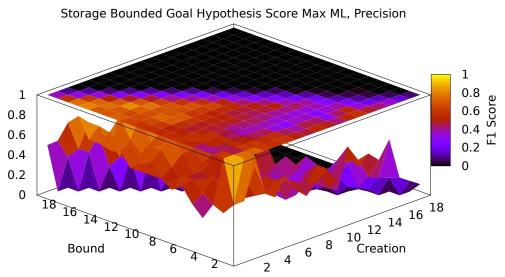


(e)

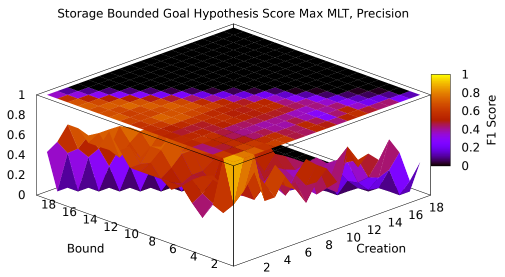


(f)

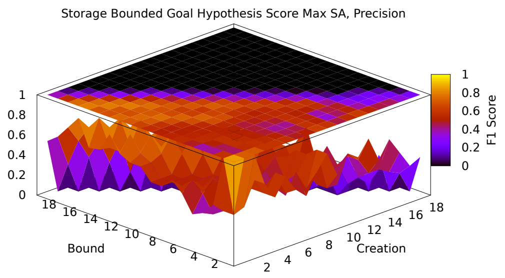
Precision bounded hypothesis results.



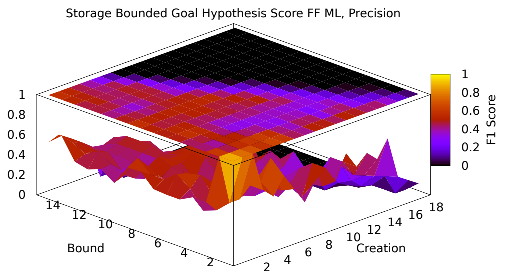
(a)



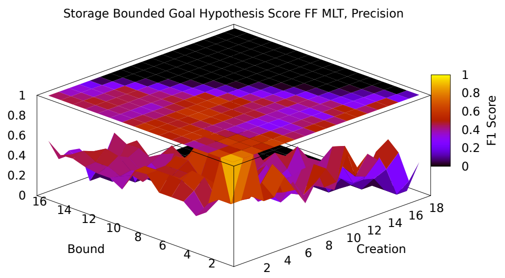
(b)



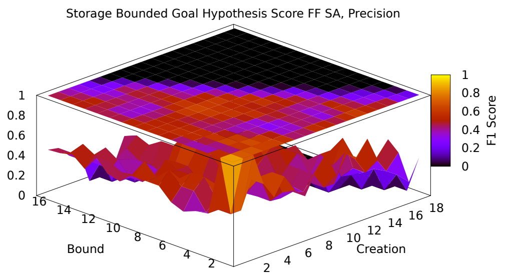
(c)



(d)

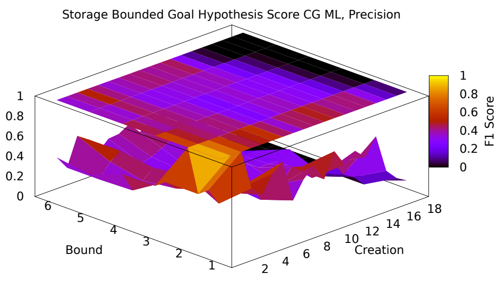


(e)

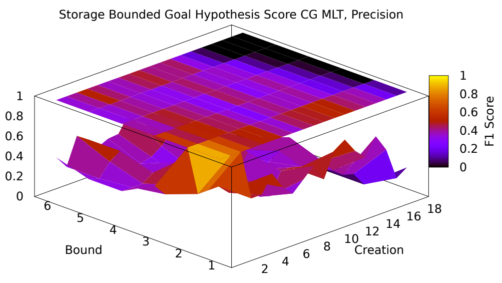


(f)

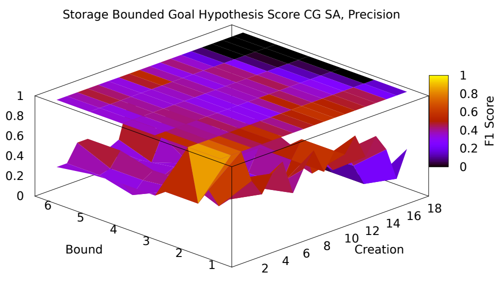
Precision bounded hypothesis results.



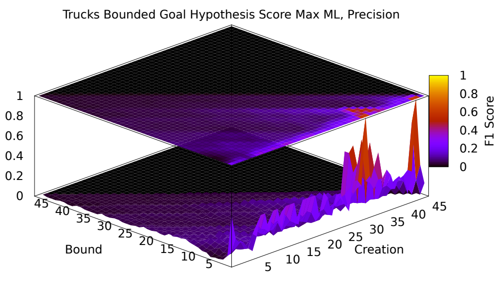
(a)



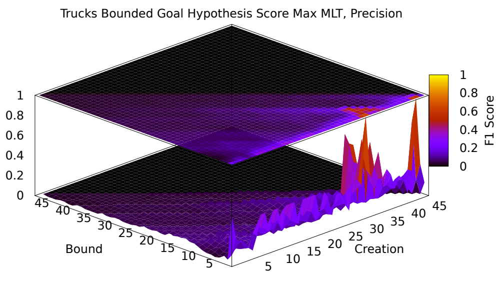
(b)



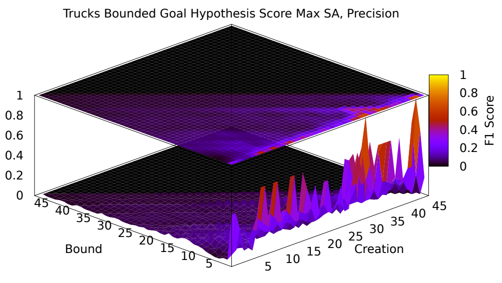
(c)



(d)

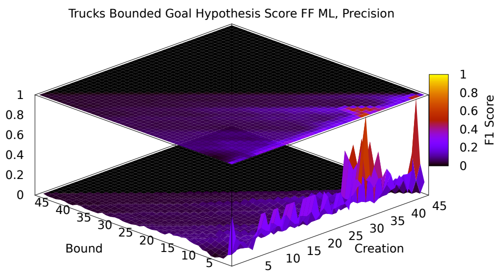


(e)

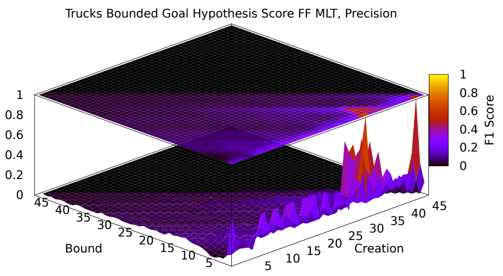


(f)

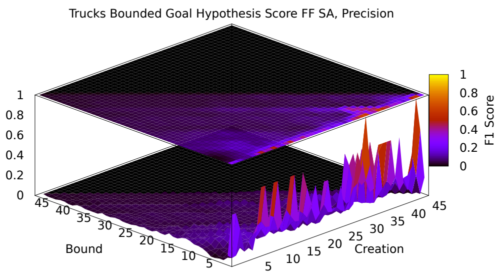
Precision bounded hypothesis results.



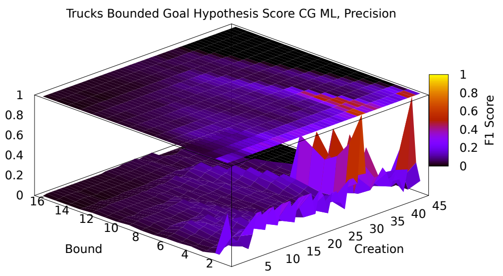
(a)



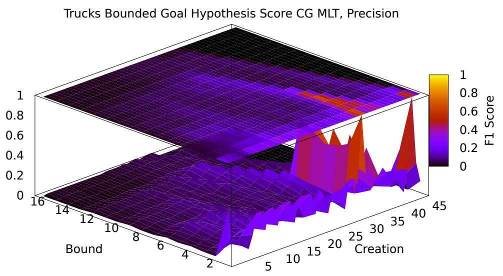
(b)



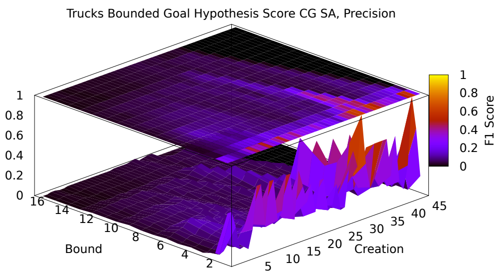
(c)



(d)



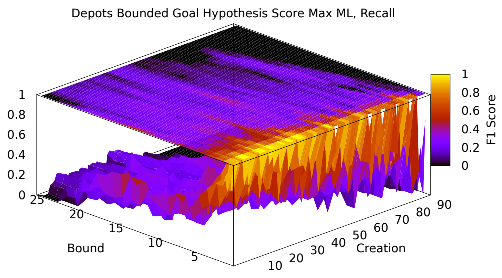
(e)



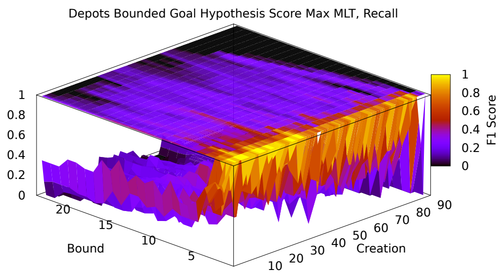
(f)

Precision bounded hypothesis results.

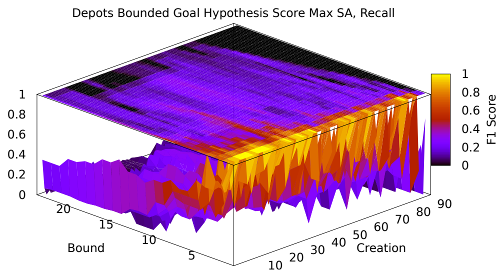
B.3 Recall Score



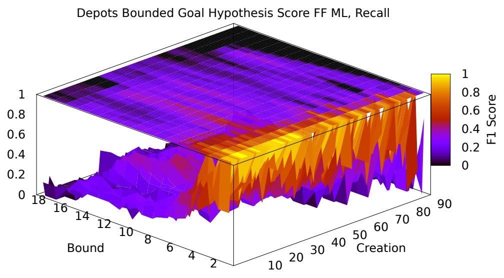
(a)



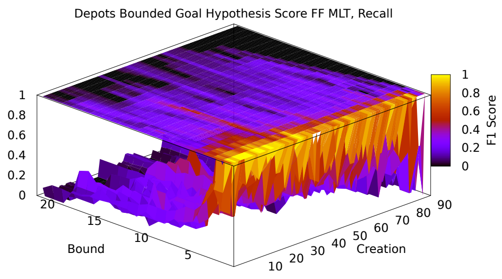
(b)



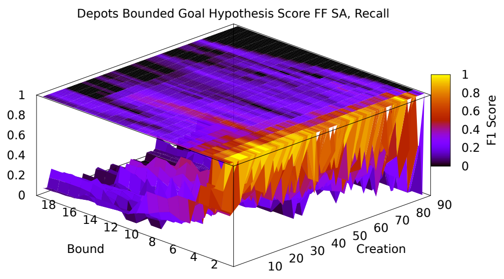
(c)



(d)

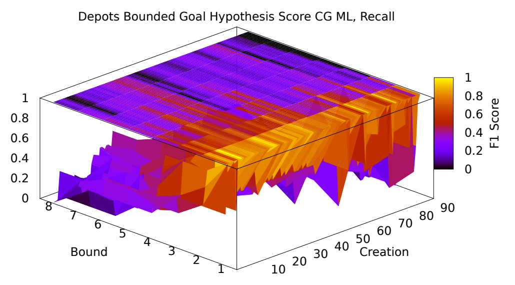


(e)

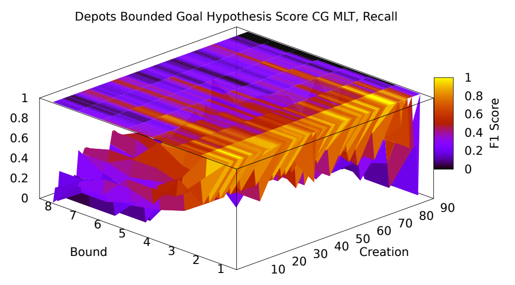


(f)

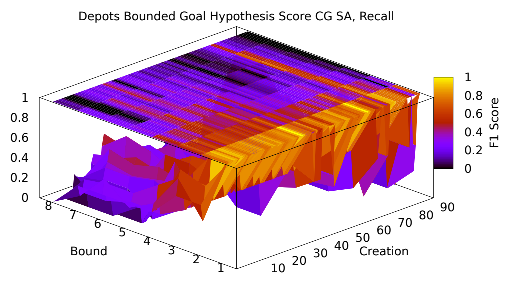
Figure B.3: *Recall* bounded hypothesis results.



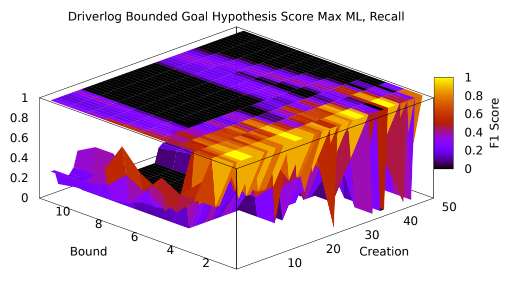
(a)



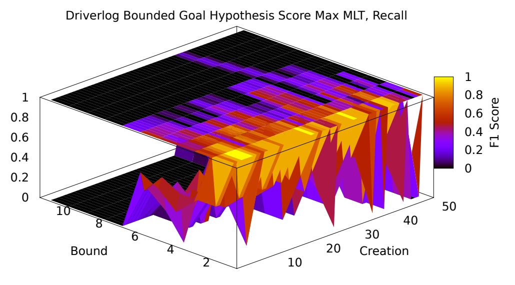
(b)



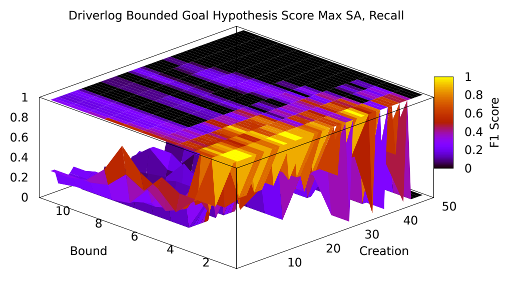
(c)



(d)

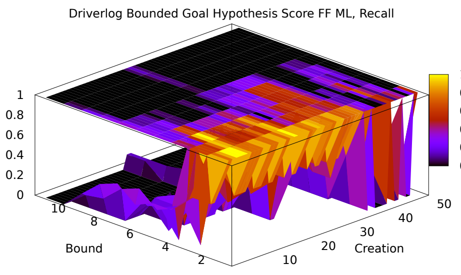


(e)

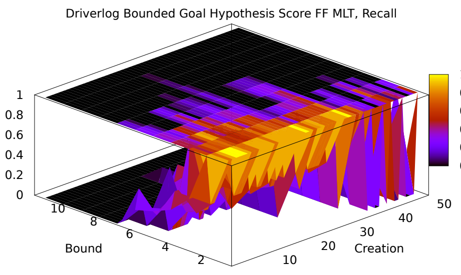


(f)

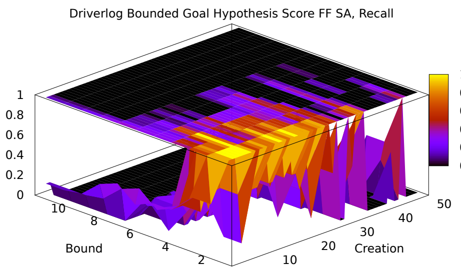
Recall bounded hypothesis results.



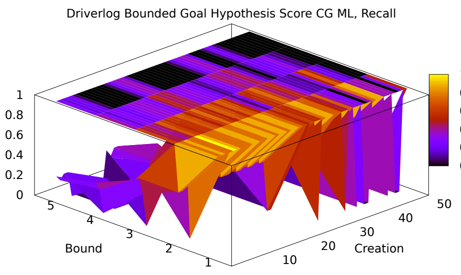
(a)



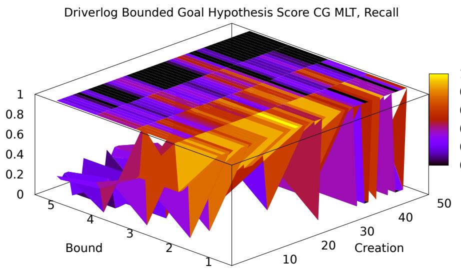
(b)



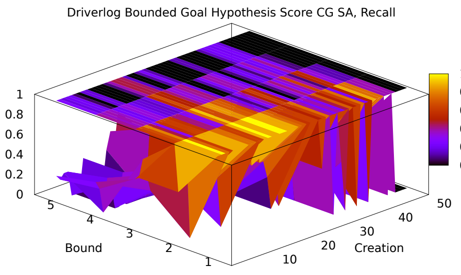
(c)



(d)

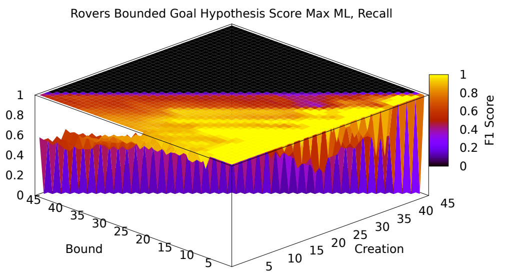


(e)

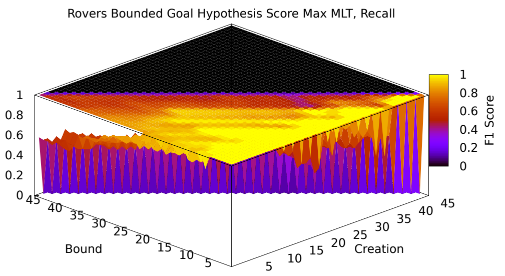


(f)

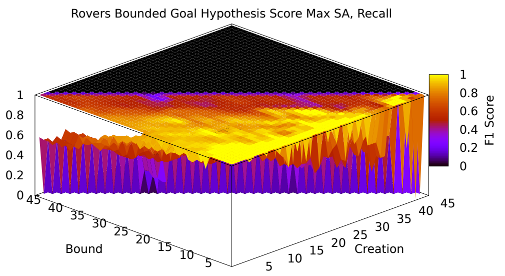
Recall bounded hypothesis results.



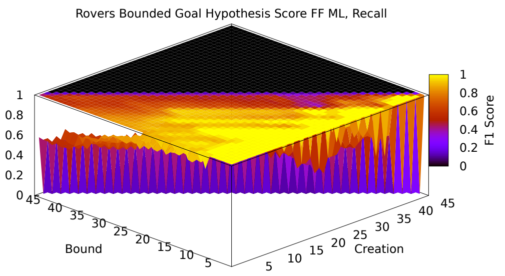
(a)



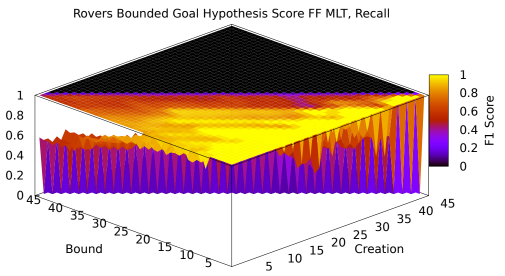
(b)



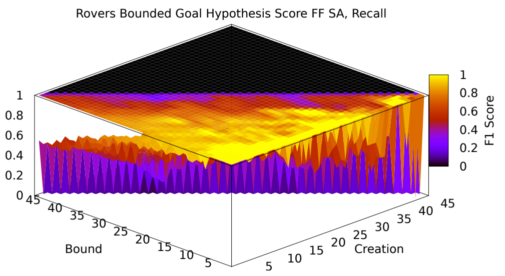
(c)



(d)

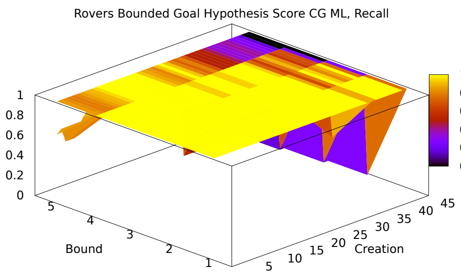


(e)

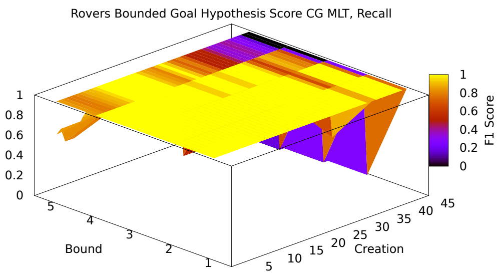


(f)

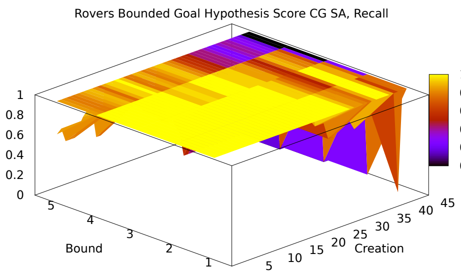
Recall bounded hypothesis results.



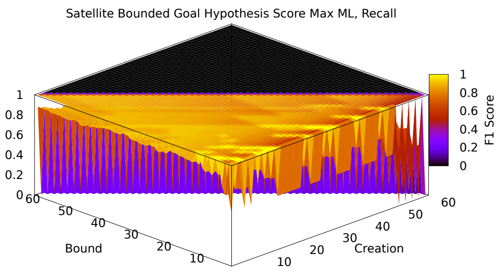
(a)



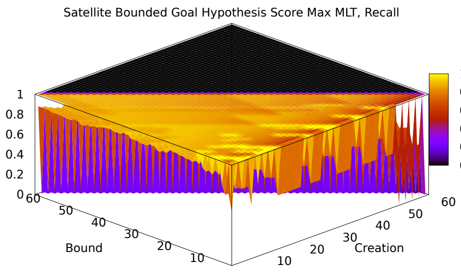
(b)



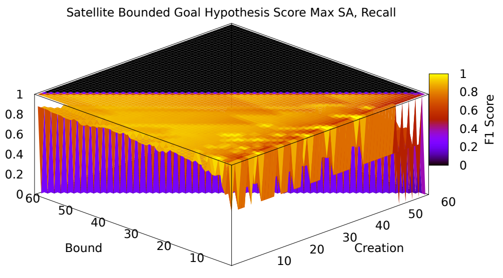
(c)



(d)

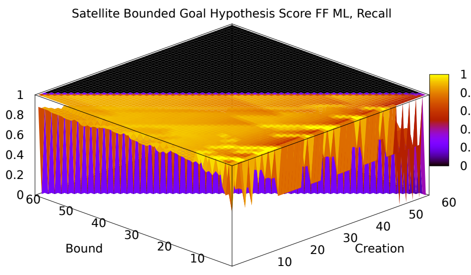


(e)

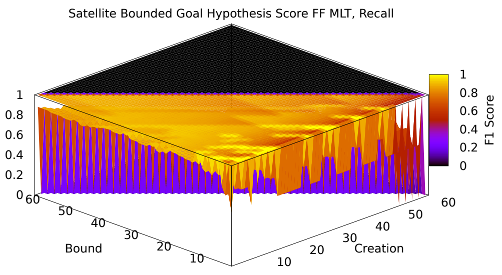


(f)

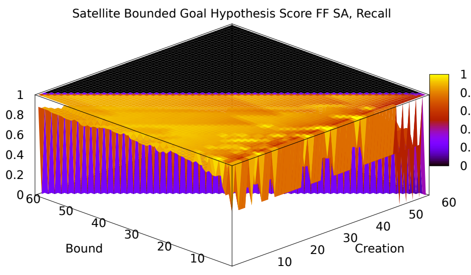
Recall bounded hypothesis results.



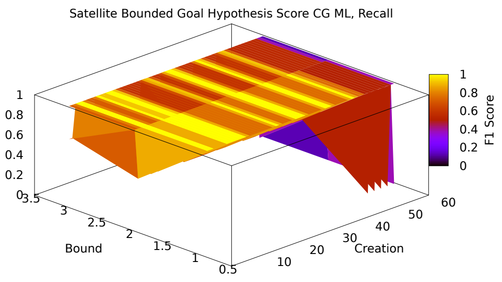
(a)



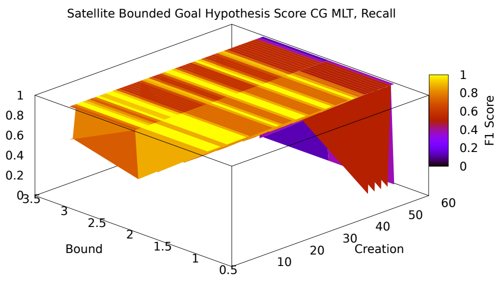
(b)



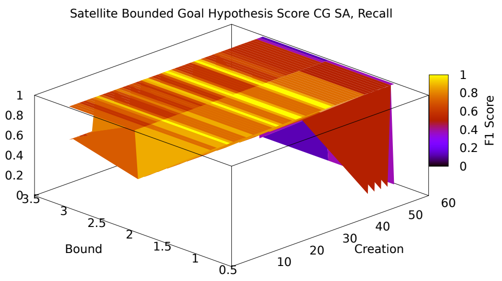
(c)



(d)

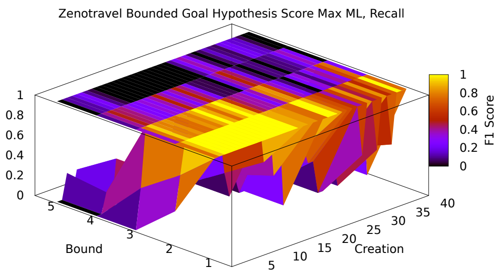


(e)

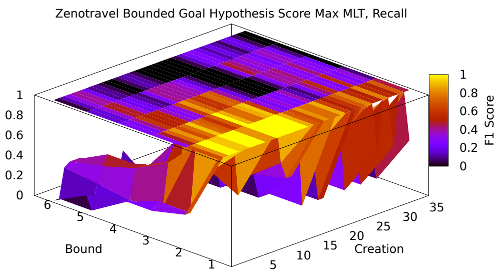


(f)

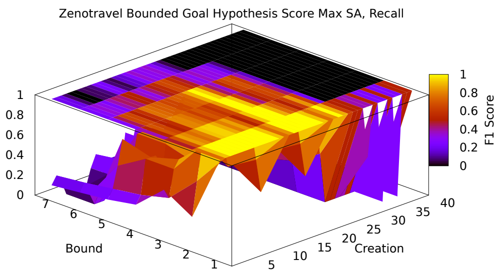
Recall bounded hypothesis results.



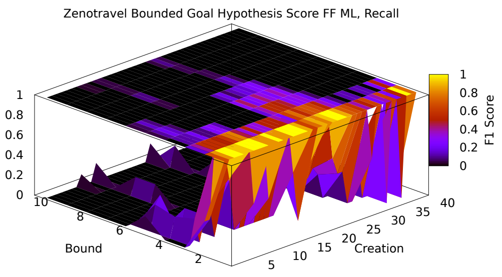
(a)



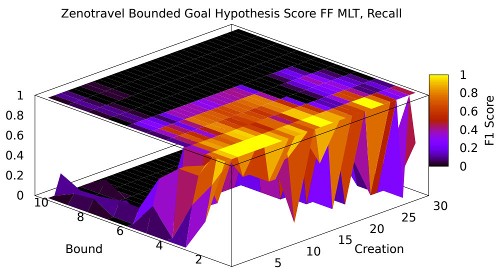
(b)



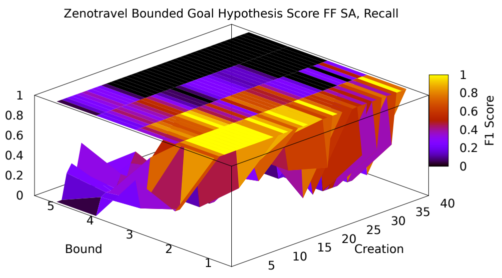
(c)



(d)

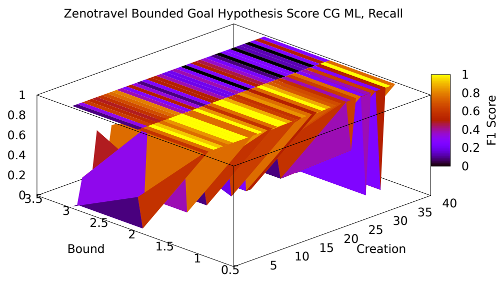


(e)

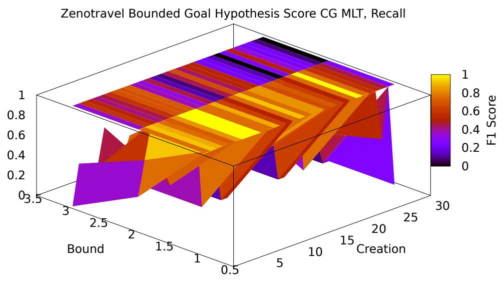


(f)

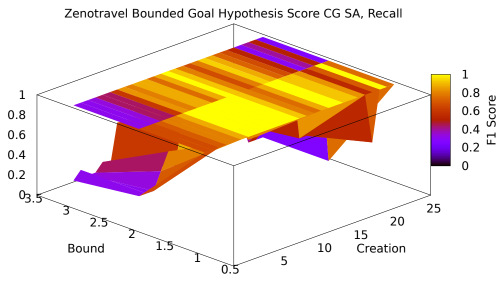
Recall bounded hypothesis results.



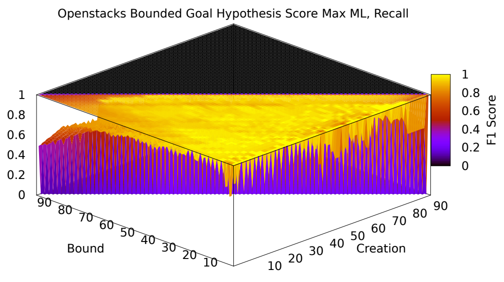
(a)



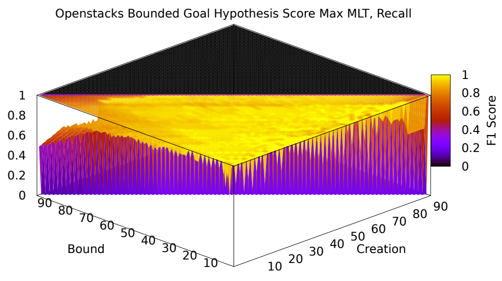
(b)



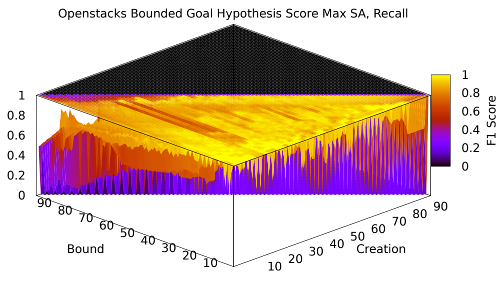
(c)



(d)

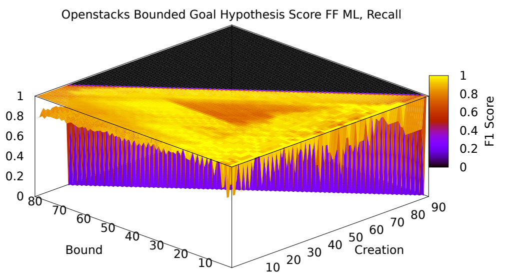


(e)

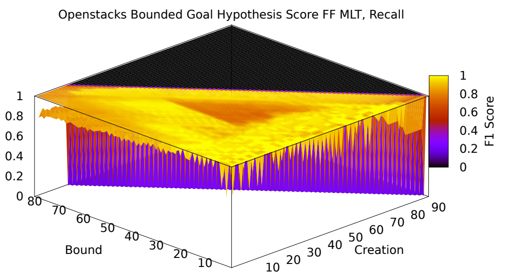


(f)

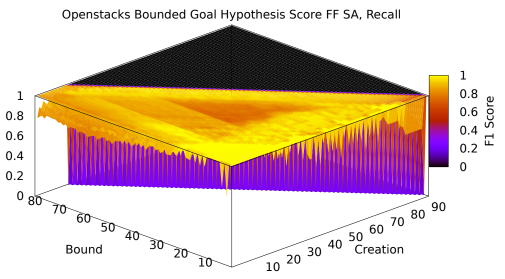
Recall bounded hypothesis results.



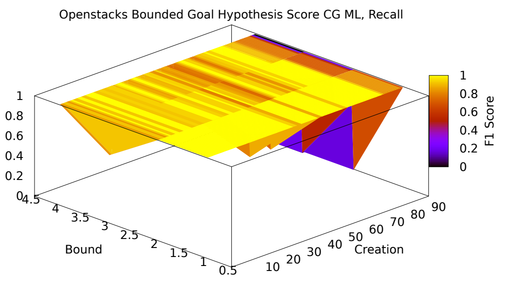
(a)



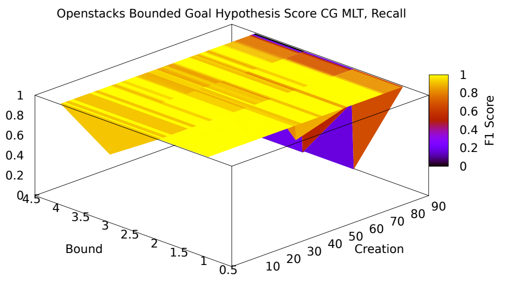
(b)



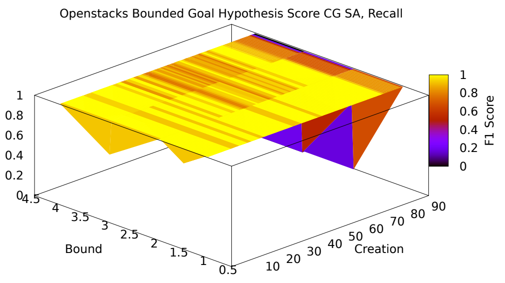
(c)



(d)

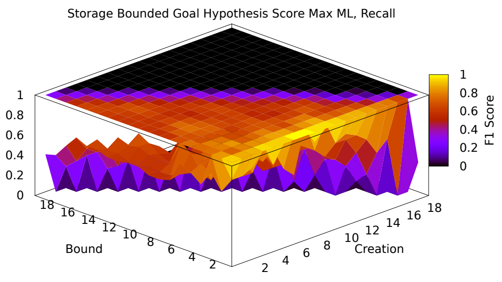


(e)

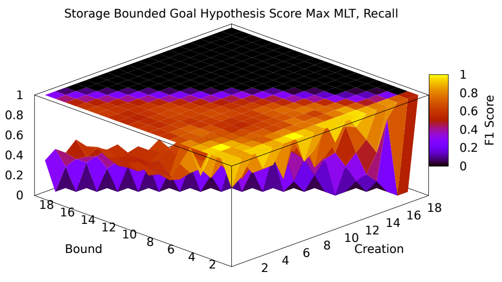


(f)

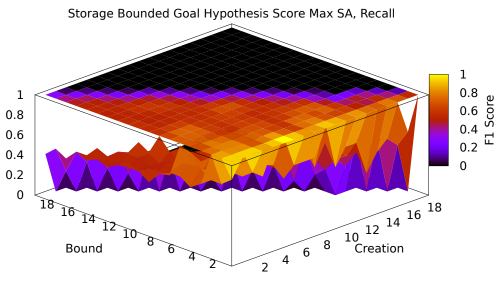
Recall bounded hypothesis results.



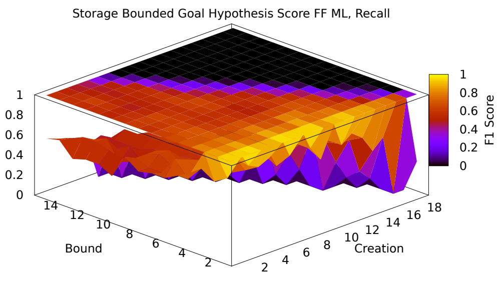
(a)



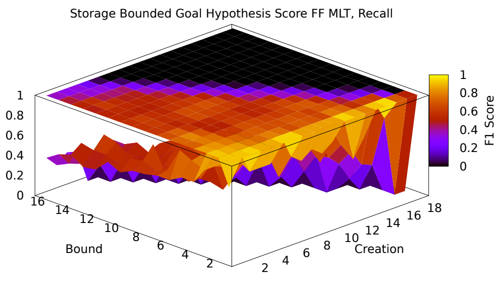
(b)



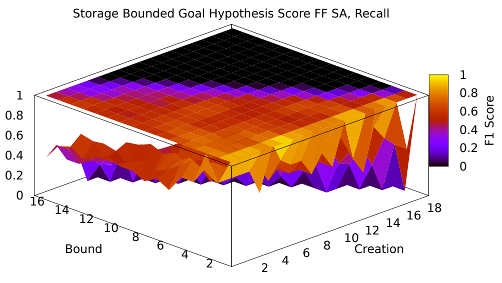
(c)



(d)

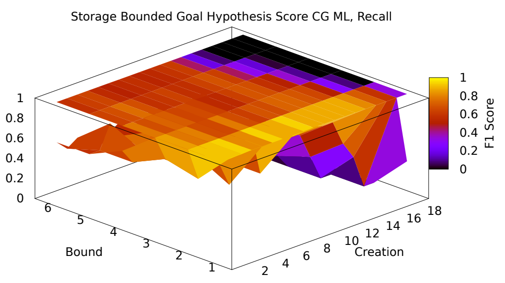


(e)

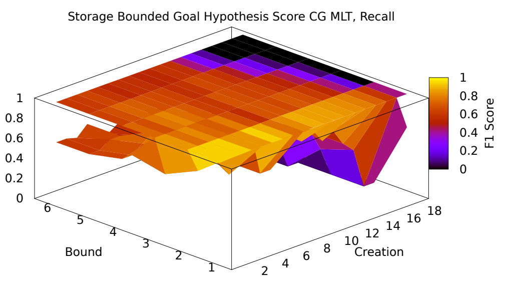


(f)

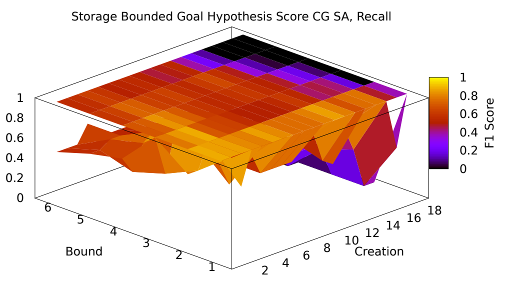
Recall bounded hypothesis results.



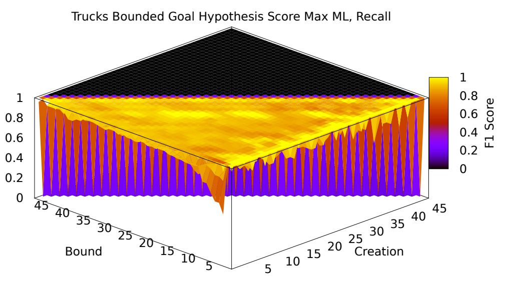
(a)



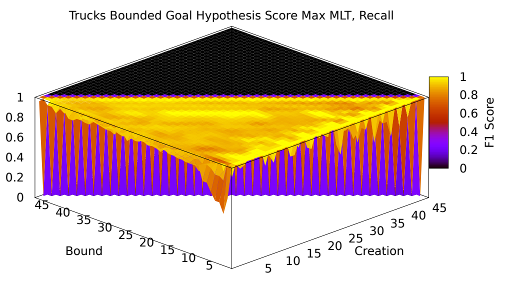
(b)



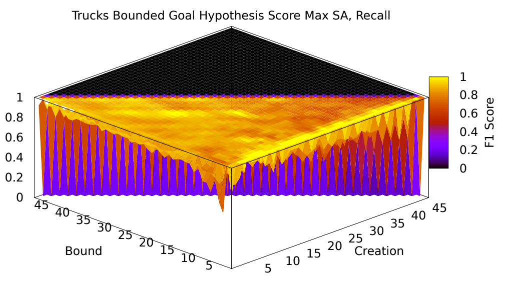
(c)



(d)

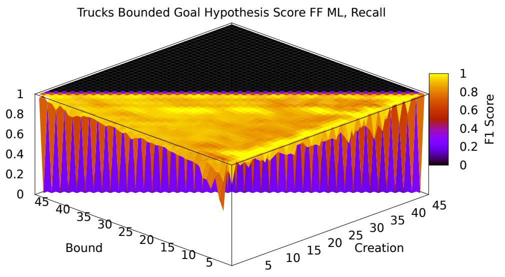


(e)

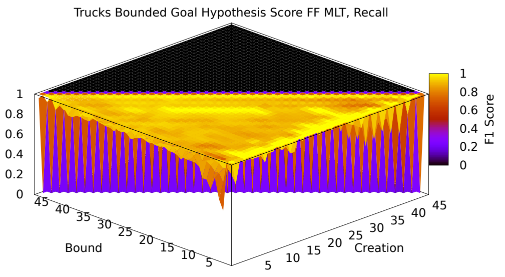


(f)

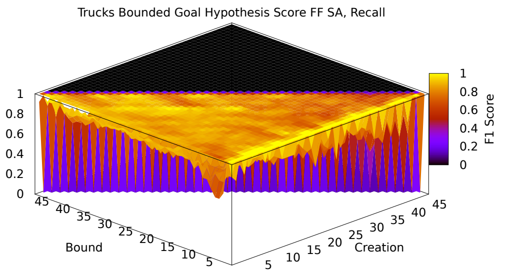
Recall bounded hypothesis results.



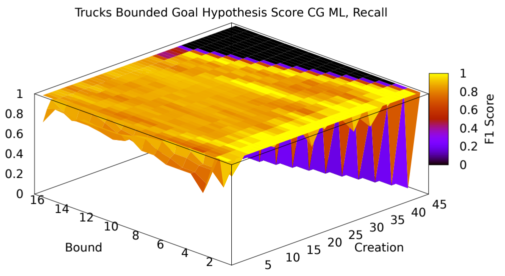
(a)



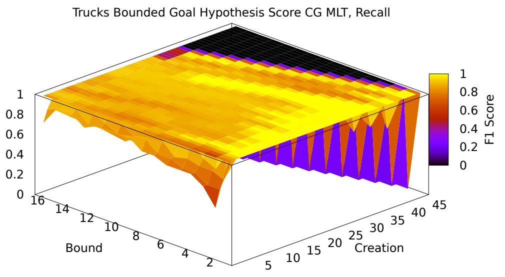
(b)



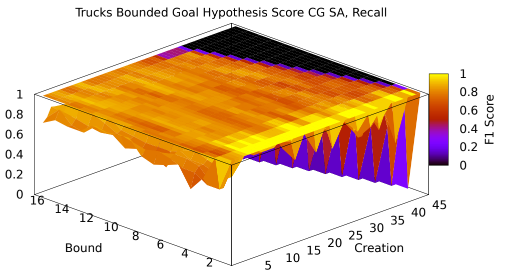
(c)



(d)



(e)



(f)

Recall bounded hypothesis results.

Appendix C

Runtime

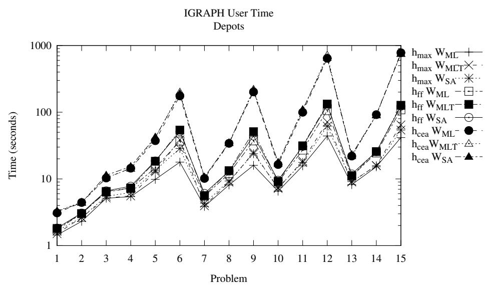
This appendix provides results on the runtime of IGRAPH, on the IPC domains tested in Chapter 5. As the system makes heavy use of heuristic calculations, these and the associated size of the goal-space dictate the runtime and resources required to perform recognition.

Section C.1 contains figures displaying the CPU time required to perform recognition on each domain and problem, while Section C.2 shows the wall-clock time for the same tests. Finally, Section C.3 shows results for the maximum memory requirements of IGRAPH.

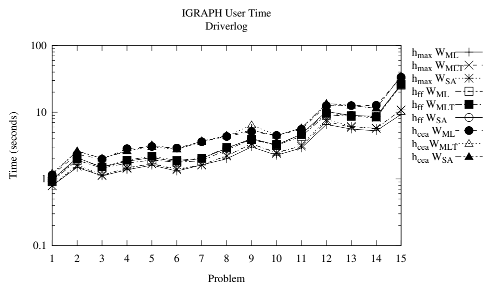
In all cases of runtime there is an exponential growth as the goal-space size increases. Fluctuations in runtime between consecutive problems (such as the CPU time results for DEPOTS) is caused by the same input problem being used for 3-4 problems, but with an increasing number of goals. As is to be expected, configurations using h_{max} have the lowest runtimes, followed by h_{ff} and h_{cea}. The work function used has no discernible impact on runtime.

Finally, the maximum amount of memory used by IGRAPH does tend to grow as problem number and goal-space size increase, but at a far lower rate than runtime. However, there can be a large difference in memory required between configurations of the system. For example, while the $\langle h_{max}, W_{ML} \rangle$ configuration requires approximately 50MB on SATELLITE, the pairing of $\langle h_{cea}, W_{SA} \rangle$ requires over 2GB.

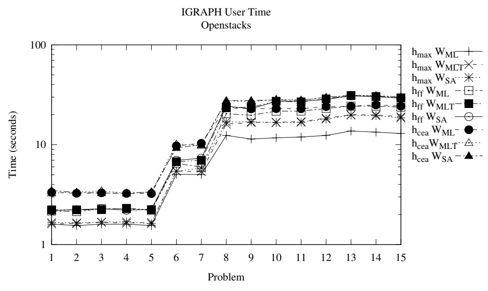
C.1 User Time



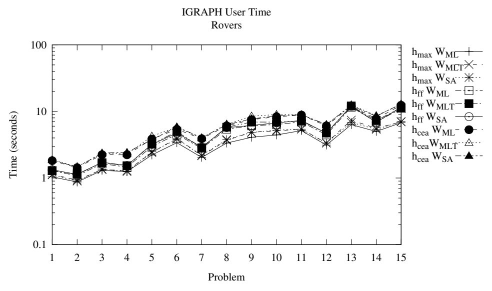
(a) Depots



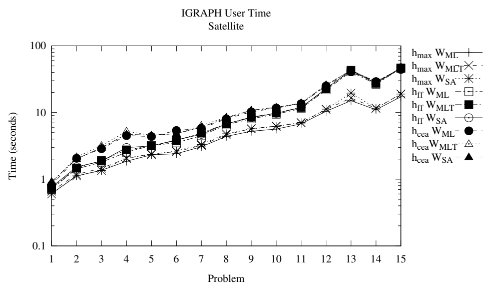
(b) Driverlog



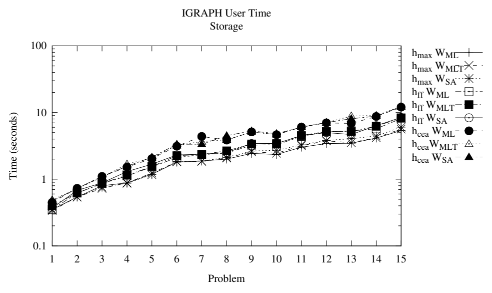
(c) Openstacks



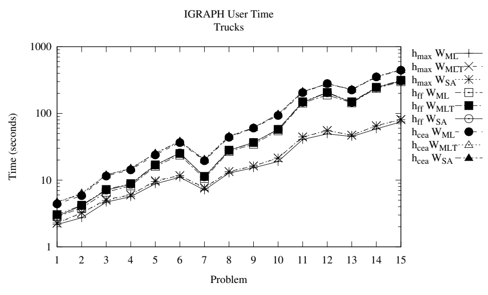
(d) Rovers



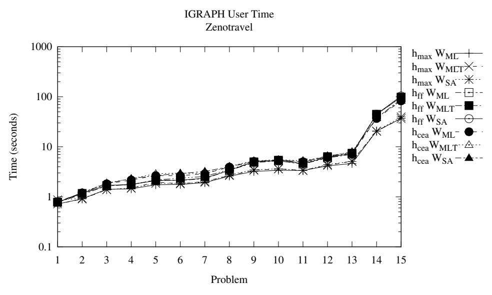
(e) Satellite



(f) Storage



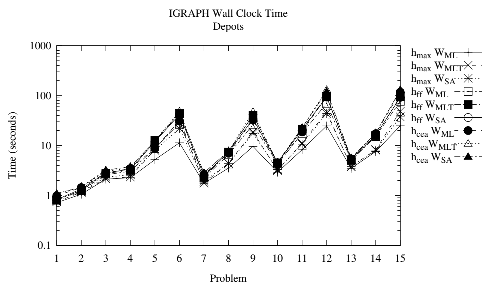
(g) Trucks



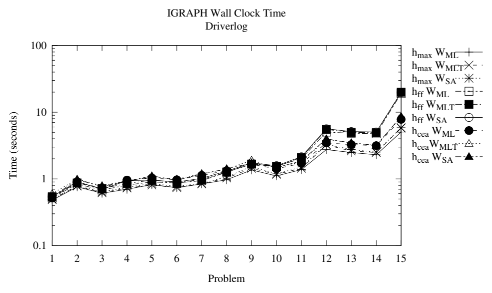
(h) Zenotravel

Figure C.1: Results for CPU time on the IPC domains.

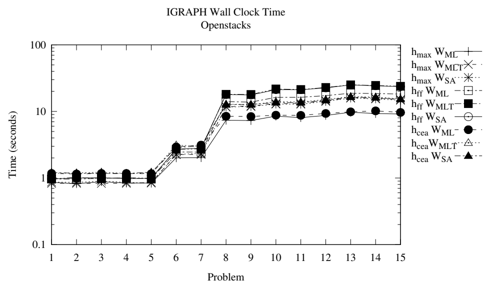
C.2 Wall-Clock Time



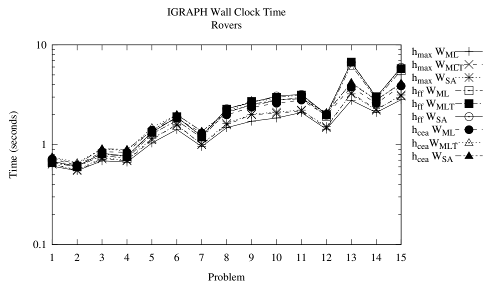
(a) Depots



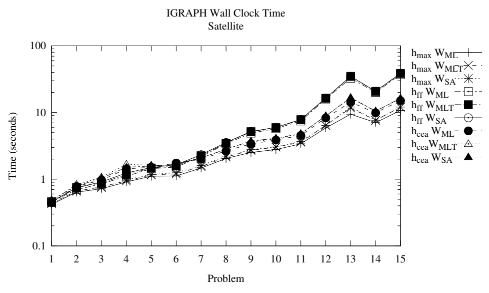
(b) Driverlog



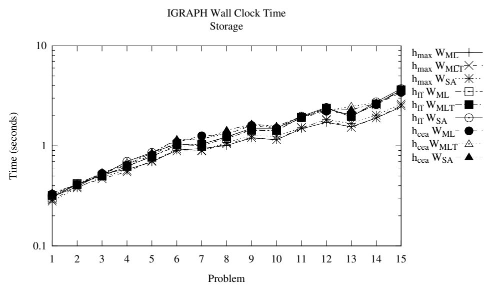
(c) Openstacks



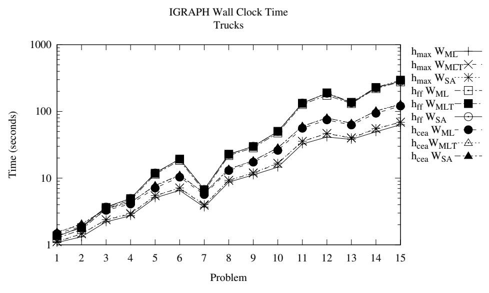
(d) Rovers



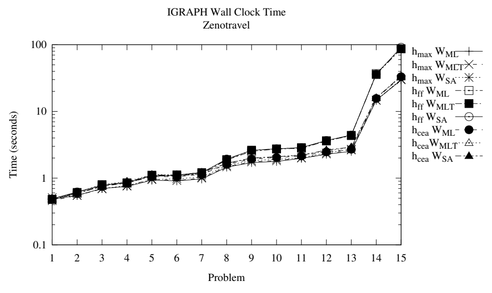
(e) Satellite



(f) Storage



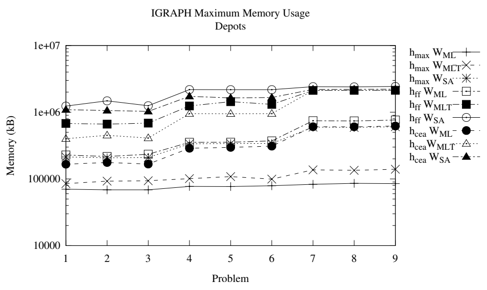
(g) Trucks



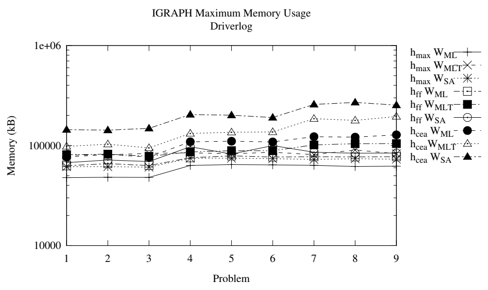
(h) Zenotravel

Figure C.2: Results for wall-clock time on the IPC domains.

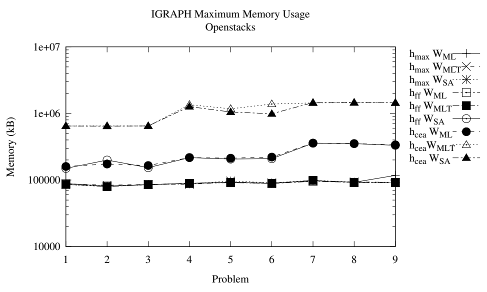
C.3 Maximum Memory Usage



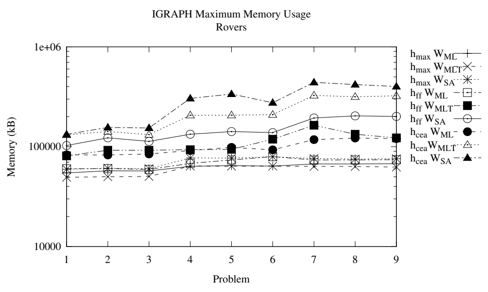
(a) DEPOTS



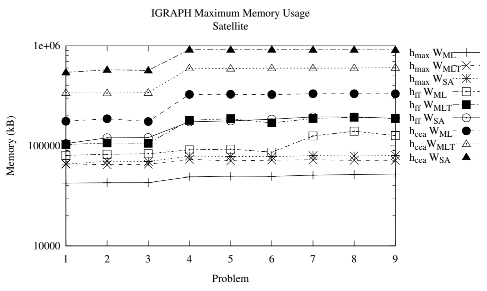
(b) DRIVERLOG



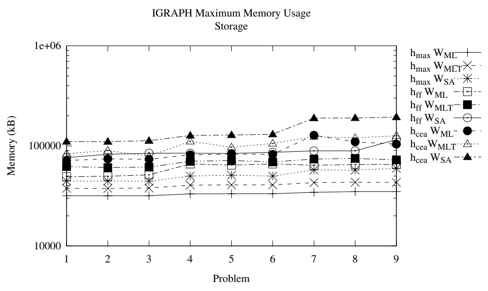
(c) OPENSTACKS



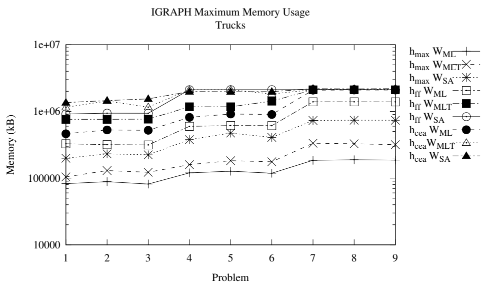
(d) ROVERS



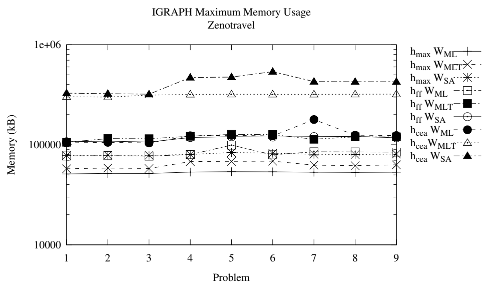
(e) SATELLITE



(f) STORAGE



(g) TRUCKS



(h) ZENOTRAVEL

Figure C.3: Results for maximum memory requirements on the IPC domains.

Appendix D

International Planning Competition Domains

D.1 Overview

This section provides an overview of the various domains used in the evaluation of IGRAPH. All of these have been taken from the 3rd and 5th International Planning Competition (IPC).

D.2 IPC3 Domains

Many of the domains in the 3rd IPC were intended to model real-life scenarios. It is perhaps expected then, that many of them model an underlying logistics problem with associated *transportation network*. Further descriptions of these domains along with results and analysis, can be found in [117].

- **Depots**

  A combined transportation and stacking domain. Packages can be moved between depots by loading them onto and off of trucks at other depots. Packages can then be stacked onto palettes or other packages, using a hoist. The problem demonstrates potential parallelism in moving trucks between depots, while the stacking problem exhibits complex goal interaction.

- **Driverlog**

  A transportation domain in which packages are delivered by trucks from locations interconnected by a road network. Drivers are required for trucks to move between locations connected by a road, but drivers can themselves walk between locations connected by paths. Often, paths between two locations are connected by a central pseudo-location, such that walking requires two actions, while driving requires one.

- **Rovers**

  This domain is based upon a number of robotic rovers completing goals on another planet such as Mars. It contains a waypoint map for traversal between objects of interest, such as rocks or soil, which can be sampled and the results communicated back to the lander. In general, there are multiple variations to this communication, such as transmitting an image in high resolution or low resolution; colour or black-and-white. However, most problems require that only a single variant be achieved.

  ROVERS is unique, in that each rover's plan rarely overlaps in any way. If rover A is equipped for imaging, while rover B is equipped for sampling soil, neither plans will interact or conflict, with the exception that only a single rover may communicate with the lander at one time. This is enforced by a stub fact being both added and deleted in the "communicate" actions, which is captured as a *pause* mutex relation.

- **Satellite**

  The SATELLITE domain is concerned primarily with scheduling rather than planning. In it, a satellite must take images of certain objects using differing instruments. This is achieved by moving the satellite into position, with some calibration of optics required.

- **Zenotravel**

  ZENOTRAVEL models the movement of passengers and aircraft between airports. Passengers embark and disembark aircraft, which can move between any airport in the problem domain. Aircraft have an associated fuel-level, which is represented as a series of literals rather than numerically. They may choose to fly normally between locations, in which a single unit of fuel is used, or can choose to "zoom" between location, in which case two units are consumed. Aircraft may refuel at airports. A single unit of fuel is added per "refuel" action.

D.3 IPC5 Domains

In the 5th IPC, many of the domains used contained more expressive syntax than IPC3. Portions of this grammar is unsupported by both the parser used in IGRAPH and the SAS+ scripts used for translation. As such, only three domains are taken from the competition.

- **Openstacks**

  Within the Openstacks domain, various products must be manufactured and combined into a single order. Each order requires a *stack* area which will be occupied by products while the entire order is incomplete. The problem lies in minimising the number of simultaneously open stacks during production, as these take up space on the shop-floor.

  The Openstacks domain is an optimisation problem, in that it is trivial to find plans which achieve the hard constraints imposed by the domain, but that it is extremely difficult to find a plan which minimises the number of open stacks, by building products in the most optimal order.

- **Storage**

  At first glance, the Storage domain is very similar to the Depots domain, in which trucks and hoists must stack crates in a specified order. However, here the crate-stacking problem is removed and replaced with the ability for the hoists which load/unload crates to move between storage and transit areas.

  As there is no ordering to crates and hoists are moveable, loose fact-ordering and tight action-ordering constraints are present. This results in a large (potentially complete) action-space.

- **Trucks**

  Trucks can be seen as a derivative of both Driverlog and Depots. Here, packages are again transported about by trucks to their destinations, with the difference that trucks no longer have unlimited capacity, and once packages arrive at their destination, they are classed as "delivered" rather than simply being "at" the location.

  The addition of limited space inside trucks has interesting implications for RPG-based heuristics, as these cannot determine that the truck may have to make the same trip several times to deliver all packages. However, the move to packages being "delivered" is beneficial for recognition as these are *strictly terminal* facts.

Bibliography

[1] David W. Albrecht, Ingrid Zukerman, and An E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8(1-2):5–47, 1998.

[2] James F. Allen, Lenhart K. Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel G. Martin, Bradford W. Miller, Massimo Poesio, and David R. Traum. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI*, 7:7–48, 1994.

[3] Han The Anh and Luis Moniz Pereira. Corpus-based incremental intention recognition via Bayesian network model construction. In *Proceedings of the First Workshop on Goal, Activity and Plan Recognition*, pages 1–8, 2011.

[4] Marcelo G. Armentano and Analía Amandi. Goal recognition with variable-order markov models. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, IJCAI'09, pages 1635–1640, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[5] Dorit Avrahami-Zilberbrand. *Efficient Hybrid Algorithms for Plan Recognition and Detection of Suspicious and Anomalous Behavior*. PhD thesis, Bar Ilan University, 2009.

[6] Dorit Avrahami-Zilberbrand and Gal A. Kaminka. Fast and complete symbolic plan recognition. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 653–658, Edinburgh, Scotland, UK, 2005. Professional Book Center.

[7] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express

search control knowledge for planning. *Artificial Intelilgence*, 116(1-2):123–191, January 2000. ISSN 0004-3702.

[8] Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp, and Malte Helmert, editors. *Proceedings of the 21st International Conference on Automated Planning and Scheduling*, 2011. AAAI.

[9] Christer Bäckström. Equivalence and tractability results for SAS$^+$ planning. In *Proceedings of Third International Conference on Principles on Knowledge Representation and Reasoning (KR-92)*, pages 126–137, 1992.

[10] Christer Bäckström. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research*, 9:99–137, 1998.

[11] Christer Bäckström and Bernhard Nebel. Complexity results for SAS$^+$ planning. *Computational Intelligence*, 11:625–656, 1995.

[12] Chris L. Baker, Rebecca R. Saxe, and Joshua B. Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the Thirty-Third Annual Conference of the Cognitive Science Society*, pages 2469–2474, 2011.

[13] Xinlong Bao and Thomas G. Dietterich. Folderpredictor: Reducing the cost of reaching the right folder. *ACM Transactions on Intelligent Systems and Technology*, 2(1):8:1–8:23, January 2011. ISSN 2157-6904.

[14] Mathias Bauer. A Dempster-Shafer approach to modeling agent preferences for plan recognition. In *Numerical Uncertainty Management in User and Student Modeling*, pages 317–348, 1995.

[15] Richard Bellman. A Markovian decision process. *Indiana University Mathematics Journal*, 6:679–684, 1957. ISSN 0022-2518.

[16] Sara Bernardini and David E Smith. Finding mutual exclusion invariants in temporal planning domains. *Proceedings of 4th Workshop on Knowledge Engineering for Planning and Scheduling, KEPS 2011*, pages 31–42, 2011.

[17] Christoph Betz and Malte Helmert. Planning with h^+ in theory and practice. In *Proceedings of the 32nd annual German conference on Advances in artificial intelligence*, KI'09, pages 9–16, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 3-642-04616-9, 978-3-642-04616-2.

[18] Nate Blaylock and James F. Allen. Corpus-based, statistical goal recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1303–1308, 2003.

[19] Nate Blaylock and James F. Allen. Generating artificial corpora for plan recognition. In *User Modeling '05*, pages 179–188, 2005.

[20] Nate Blaylock and James F. Allen. Fast hierarchical goal schema recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 796–801, 2006.

[21] Nathan James Blaylock. *Towards tractable agent-based dialogue*. PhD thesis, Rochester, NY, USA, 2005. AAI3189105.

[22] Avrim Blum and John Langford. Probabilistic planning in the graphplan framework. In Susanne Biundo and Maria Fox, editors, *ECP*, volume 1809 of *Lecture Notes in Computer Science*, pages 319–332. Springer, 1999. ISBN 3-540-67866-2.

[23] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:1636–1642, 1995.

[24] Mark S. Boddy, Maria Fox, and Sylvie Thiébaux, editors. *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, 2007. AAAI. ISBN 978-1-57735-344-7.

[25] Blai Bonet and Hector Geffner. GPT: A tool for planning with uncertainty and partial information. In *Proceedings of Workshop on Planning with Uncertainty and Incomplete Information, IJCAI 2001*, pages 82–87, 2001.

[26] Blai Bonet and Hector Geffner. Heuristic search planner 2.0. *AI Magazine*, 22(3):77–80, 2001.

[27] Blai Bonet and Hector Geffner. Planning as heuristic search. *Journal of Artificial Intelligence Research*, 129(1-2):5–33, 2001.

[28] Blai Bonet, Gábor Loerincs, and Hector Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of AAAI-97*, pages 714–719. MIT Press, 1997.

[29] Sviatoslav Braynov. *Adversarial Planning in Networks*, pages 263–274. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01140-5.

[30] Hung H. Bui. A general model for online probabilistic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1309–1315, 2003.

[31] Hung H. Bui, Svetha Venkatesh, and Geoff West. Policy recognition in the abstract hidden Markov model. *Journal of Artificial Intelligence Research*, 17(1):451–499, December 2002. ISSN 1076-9757.

[32] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:165–204, September 1994. ISSN 0004-3702.

[33] Sandra Carberry. Incorporating default inference into plan recognition. In *AAAI-90*, 1990.

[34] Sandra Carberry. *Plan Recognition in Natural Language Dialogue.* MIT Press, Cambridge, MA, USA, 1990. ISBN 0262031671.

[35] Sandra Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001.

[36] Eugene Charniak. *Statistical Language Learning.* MIT Press, Cambridge, MA, USA, 1994. ISBN 0262032163.

[37] Eugene Charniak and Robert P. Goldman. A probabilistic model of plan recognition. In *AAAI*, pages 160–165, 1991.

[38] Eugene Charniak and Robert P. Goldman. A Bayesian model of plan recognition. *Journal of Artificial Intelligence Research*, 64(1):53–79, 1993.

[39] Yixin Chen, Benjamin W. Wah, and Chih-Wei Hsu. Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research*, 26:323–369, 2006.

[40] Yixin Chen, You Xu, and Guohui Yao. Stratified planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, pages 1665–1670, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[41] P.R. Cohen, C.R. Perrault, and J.F. Allen. *Beyond question answering.* BBN Report. Bolt Beranek and Newman, 1981.

[42] A. I. Coles, M. Fox, D. Long, and A. J. Smith. A hybrid relaxed planning graph-lp heuristic for numeric planning domains. In *Proceedings of the International Conference on Automated Planning and Scheduling*, September 2008.

[43] A. J. Coles, A. I. Coles, M. Fox, and D. Long. Forward-chaining partial-order planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, May 2010.

[44] Andrew Coles, Maria Fox, Derek Long, and Amanda Smith. Teaching forward-chaining planning with JavaFF. In *Colloquium on AI Education, 23rd AAAI Conference on Artificial Intelligence*, 2008.

[45] Stephen Cresswell and Peter Gregory. Generalised domain model acquisition from action traces. In Bacchus et al. [8], pages 42–59.

[46] Rina Dechter. *Constraint processing.* Elsevier Morgan Kaufmann, 2003. ISBN 978-1-55860-890-0.

[47] Stefan Edelkamp and Jörg Hoffmann. PDDL2.2: The language for the Classical part of the 4th International Planning Competition. Technical Report 195, January 2004.

[48] George M. Ferguson. Explicit representation of events, actions and plans for assumption-based plan reasoning. Technical report, Rochester, NY, USA, 1992.

[49] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 608–620, 1971.

[50] Jeff Forbes, Tim Huang, Keiji Kanazawa, and Stuart Russell. The BATmobile: Towards a Bayesian automated taxi, 1995.

[51] Maria Fox and Derek Long. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research*, 9:367–421, 1998.

[52] Maria Fox and Derek Long. The detection and exploitation of symmetry in planning problems. In *International Joint Conference on Artificial Intelligence*, pages 956–961. Morgan Kaufmann, 1999.

[53] Maria Fox and Derek Long. The detection and exploitation of symmetry in planning problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 99, pages 956–961, 1999.

[54] Maria Fox and Derek Long. Utilizing automatically inferred invariants in graph construction and search. In *In Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pages 102–111. AAAI Press, 2000.

[55] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[56] Christopher Geib and Robert Goldman. Recognizing plans with loops represented in a lexicalized grammar. In *AAAI Conference on Artificial Intelligence*, 2011.

[57] Christopher Geib and Christopher Swetenham. Parallelizing plan recognition. In *AAAI Workshop: Plan, Activity, and Intent Recognition*, volume WS-13-13 of *AAAI Workshops*. AAAI, 2013.

[58] Christopher W. Geib. Delaying commitment in plan recognition using combinatory categorial grammars. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1702–1707, 2009.

[59] Christopher W. Geib and Robert P. Goldman. Probabilistic plan recognition for hostile agents. In Ingrid Russell and John F. Kolen, editors, *FLAIRS Conference*, pages 580–584. AAAI Press, 2001. ISBN 1-57735-133-9.

[60] Christopher W. Geib and Robert P. Goldman. Plan recognition in intrusion detection systems. In *DARPA Information Survivability Conference and Exposition II, 2001. DISCEX '01. Proceedings*, volume 1, pages 46–55, 2001.

[61] Christopher W. Geib and Robert P. Goldman. Recognizing plan/goal abandonment. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1515–1517. Morgan Kaufman, 2003.

[62] Christopher W. Geib and Mark Steedman. On natural language processing and plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1612–1617, 2007.

[63] Christopher W. Geib, John Maraist, and Robert P. Goldman. A new probabilistic plan recognition algorithm based on string rewriting. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 91–98, 2008.

[64] Alfonso Gerevini and Lenhart Schubert. Inferring state constraints for domain-independent planning. In *AAAI/IAAI*, pages 905–912, 1998.

[65] Alfonso Gerevini and Lenhart K Schubert. Discovering state constraints in discoplan: Some new results. In *AAAI/IAAI*, pages 761–767, 2000.

[66] Alfonso Gerevini and Ivan Serina. LPG: A planner based on local search for planning graphs with action costs. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pages 13–22. AAAI, April 23–27 2002. ISBN 1-57735-142-8.

[67] Alfonso E. Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009. ISSN 0004-3702.

[68] Model Abigail Gertner, Abigail S. Gertner, Cristina Conati, and Kurt Vanlehn. Procedural help in Andes: Generating hints using a Bayesian network student. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 106–111. AAAI Press, 1998.

[69] Robert P. Goldman and Eugene Charniak. Dynamic construction of belief networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 90–97, 1990.

[70] Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A new model of plan recognition. *Journal of Artificial Intelligence Research*, 64:53–79, 1999.

[71] Robert P. Goldman, Christopher W. Geib, Henry Kautz, and Tamim Asfour. Plan Recognition (Dagstuhl Seminar 11141). *Dagstuhl Reports*, 1(4): 1–22, 2011. ISSN 2192-5283.

[72] R.C. González and M.G. Thomason. *Syntactic Pattern Recognition: An Introduction*. Addison-Wesley Series in Sociology. Addison-Wesley Publishing Company, Advanced Book Program, 1978. ISBN 9780201029307.

[73] Peter Gregory, Derek Long, Craig McNulty, and Susanne M Murphy. Exploiting path refinement abstraction in domain transition graphs. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*, pages 971–976, 2011.

[74] Eunyoung Ha, Jonathan P. Rowe, Bradford W. Mott, and James C. Lester. Goal recognition with Markov logic networks for player-adaptive games. In Vadim Bulitko and Mark O. Riedl, editors, *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. The AAAI Press, 2011.

[75] B. Harrison, S. Ware, M. Fendt, and D. Roberts. A survey and analysis of techniques for player behavior prediction in massively multiplayer online role-playing games. *IEEE Transactions on Emerging Topics in Computing*, PP(99):1–16, 2014. ISSN 2168-6750.

[76] Patrik Haslum. Additive and reversed relaxed reachability heuristics revisited. *Proceedings of the 6th International Planning Competition*, 2008.

[77] Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock, editors, *International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pages 140–149. AAAI, 2000. ISBN 1-57735-111-8.

[78] Malte Helmert. A planning heuristic based on causal graph analysis. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 161–170, 2004.

[79] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[80] Malte Helmert and Hector Geffner. Unifying the causal graph and additive heuristics. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling, ICAPS 2008*, pages 140–147, 2008.

[81] Malte Helmert and Gabriele Röger. How good is almost perfect. In *In ICAPS-Workshop on Heuristics for Domain-Independent Planning*, 2007.

[82] Jörg Hoffmann. Local search topology in planning benchmarks: an empirical analysis. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, volume 1, pages 453–458, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-812-5, 978-1-558-60812-2.

[83] Jörg Hoffmann. The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20(1):291–341, December 2003. ISSN 1076-9757.

[84] Jörg Hoffmann. Analyzing Search Topology Without Running Any Search: On the Connection Between Causal Graphs and h+. *J. AI. Res.*, 41:155–229, 2011.

[85] Jörg Hoffmann and Ronen I. Brafman. Contingent planning via heuristic forward search with implicit belief states. In Susanne Biundo, Karen L. Myers, and Kanna Rajan, editors, *ICAPS*, pages 71–80. AAAI, 2005.

[86] Jörg Hoffmann and Ronen I. Brafman. Conformant planning via heuristic forward search: A new approach. *Artificial Intelilgence*, 170(6–7):507–541, 2006.

[87] Jörg Hoffmann and Ronen I. Brafman. Conformant planning via heuristic forward search: a new approach. *Artificial Intelilgence*, 170:507–541, May 2006. ISSN 0004–3702.

[88] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[89] Jun Hong. Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research*, 15:1–30, 2001.

[90] Jeffrey D. Hopcroft, John E.; Ullman. *Introduction to Automata Theory, Languages, and Computation*, chapter 4: Context-Free Grammars, pages 77–106. Addison-Wesley, 1979.

[91] Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265. Morgan Kaufmann, 1998.

[92] Marcus Huber, Edmund H. Durfee, and Michael P. Wellman. The automated mapping of plans for plan recognition. In *Proceedings of 10th Conference on Uncertainty in Artificial Intelligence*, pages 344–351. Morgan Kaufmann, 1994.

[93] Alexander Huntemann, Eric Demeester, Hendrik van Brussel, and Marnix Nuttin. Can Bayes help disabled users? A Bayesian approach to plan recognition and shared control for steering an electrical wheelchair. In *Proceedings ACM/IEEE Human-Robot Interaction Conference*, pages 67–70, 2008.

[94] Peter A. Jarvis, Teresa F. Lunt, and Karen L. Myers. Identifying terrorist activity with ai plan-recognition technology. *AI Magazine*, 2005.

[95] Froduald Kabanza, Philipe Bellefeuille, Francis Bisson, Abder Rezak Benaskeur, and Hengameh Irandoust. Opponent behaviour recognition for real-time strategy games. In *Plan, Activity, and Intent Recognition*, volume WS-10-05 of *AAAI Workshops*. AAAI, 2010.

[96] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artifical Intelligence*, 101:99–134, 1998.

[97] Henry A. Kautz. *A formal theory of plan recognition*. PhD thesis, University of Rochester, 1987.

[98] Henry A. Kautz and James F. Allen. Generalized plan recognition. In *Proceedings of AAAI-86*, pages 32–37, 1986.

[99] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design. In Steve Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI, 2014. ISBN 978-1-57735-660-8.

[100] Craig Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68:243–302, 1994.

[101] Jana Koehler. Solving complex planning tasks through extraction of subproblems. In *In Proceedings 4th International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pages 62–69. AAAI Press, Menlo Park, 1998.

[102] Jana Koehler and Jörg Hoffmann. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *Journal of Artificial Intelligence Research*, 12(1):339–386, June 2000. ISSN 1076-9757.

[103] Jana Koehler and Jörg Hoffmann. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *Journal of Artificial Intelligence Research*, 12(1):339–386, 2000.

[104] Alexander Kott, Rajdeep Singh, William M. McEneaney, and Wes Milks. Hypothesis-driven information fusion in adversarial, deceptive environments. *Information Fusion*, 12:131–144, April 2011. ISSN 1566-2535.

[105] Peter Krauthausen and Uwe D. Hanebeck. Intention recognition for partial-order plans using dynamic Bayesian networks. In *FUSION'09*, pages 444–451, 2009.

[106] Jonas Kvarnstrm and Patrick Doherty. TALplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):119–169, 2000. ISSN 1012-2443.

[107] F. De la Torre, J. Hodgins, J. Montano, S. Valcarcel, R. Forcada, and J. Macey. Guide to the carnegie mellon, university multimodal activity (cmu-mmac) database, mu-ri-tr-08-22. Technical report, Robotics Institute, Carnegie Mellon University, July 2009.

[108] Kennard Laviers and Gita Sukthankar. A real-time opponent modeling system for Rush football. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2476–2481, Barcelona, Spain, July 2011.

[109] Neal Lesh. Scalable and adaptive goal recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1208–1214, 1997.

[110] Neal Lesh. *Scalable and Adaptive Goal Recognition*. PhD thesis, University of Washington, 1998.

[111] Neal Lesh and Oren Etzioni. A sound and fast goal recognizer. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1704–1710, 1995.

[112] Neal Lesh and Oren Etzioni. Scaling up goal recognition. In *Knowledge Representation*, pages 244–255, 1996.

[113] Neal Lesh, Charles Rich, Candace L. Sidner, and Ace L. Sidner. Using plan recognition in human-computer collaboration. In *In Proceedings of the Seventh International Conference on User Modeling*, pages 23–32, 1999.

[114] Steven Levine and Brian Williams. Concurrent plan recognition and execution for human-robot teams. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[115] Nir Lipovetzky and Hector Geffner. Searching for plans with carefully designed probes. In Bacchus et al. [8].

[116] Nir Lipovetzky and Hector Geffner. Width and serialization of classical planning problems. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 540–545. IOS Press, 2012. ISBN 978-1-61499-097-0.

[117] Derek Long and Maria Fox. The third international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59, 2003.

[118] B. Majecka. Statistical models of pedestrian behaviour in the forum. Master's thesis, School of Informatics, University of Edinburgh, 2009.

[119] Drew McDermott. Using regression-match graphs to control search in planning. *Artificial Intelligence*, 109:111–159, 1999.

[120] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL - the planning domain definition language. Technical report, Yale Center for Computational Vision and Control, 1998.

[121] Vitaly Mirkis and Carmel Domshlak. Cost-sharing approximations for h+. In Boddy et al. [24], pages 240–247. ISBN 978-1-57735-344-7.

[122] Bradford Mott, Sunyoung Lee, and James Lester. Probabilistic goal recognition in interactive narrative environments. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 187–192, 2006.

[123] K. P. Murphy and Mark A. Paskin. Linear-time inference in hierarchical HMMs. In *Proceedings of Neural Information Processing Systems, NIPS 2001*, pages 833–840, 2001.

[124] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. University of California, Berkeley, 2002.

[125] Dana Nau, Malik Ghallab, and Paolo Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 1558608567.

[126] Dana S. Nau, Yue Cao, Amnon Lotem, and Héctor Muñoz-Avila. The SHOP planning system. *AI Magazine*, 22(3):91–64, 2001.

[127] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[128] Héctor Palacios and Hector Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35:623–675, 2009.

[129] David Pattison and D. Long. Accurately determining intermediate and terminal plan states using Bayesian goal recognition. In David Pattison, Derek Long, and Christopher W. Geib, editors, *Proceedings of the First*

Workshop on Goal, Activity and Plan Recognition (GAPRec), pages 32 – 37, June 2011.

[130] David Pattison and Derek Long. Domain independent goal recognition. In *STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium*, volume 222, pages 238 – 250. IOS Press, August 2010.

[131] David Pattison and Derek Long. Extracting plans from plans. In S. Fratini, A. Gerevini, D. Long, and A. Saetti, editors, *Proceedings of the 28th Workshop of the UK Special Interest Group on Planning and Scheduling, PLANSIG'10*, pages 149 – 156, December 2010.

[132] David Pattison, Derek Long, and Christopher W. Geib. A plan recognition competition – are we ready? In *Proceedings of the First Workshop on Goal, Activity and Plan Recognition (GAPRec)*, pages III – VI, 2011.

[133] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Representation and Reasoning Series. Morgan Kaufmann, 1997. ISBN 9781558604797.

[134] Edwin P. D. Pednault. ADL: exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the first international conference on Principles of knowledge representation and reasoning*, pages 324–332, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1-55860-032-9.

[135] C. Raymond Perrault and James F. Allen. A plan-based analysis of indirect speech acts. *Computational Linguistics*, 6(3-4):167–182, July 1980. ISSN 0891-2017.

[136] David Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

[137] Julie Porteous and Laura Sebastia. Extracting and ordering landmarks for planning. In *Proceedings UK Planning and Scheduling SIG Workshop*, 2000.

[138] Julie Porteous, Laura Sebastia, and Jörg Hoffmann. On the extraction, ordering, and usage of landmarks in planning. In *Proceedings of the European Conference on Artificial Intelligence*, pages 37–48, 2001.

[139] David V. Pynadath and Michael P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 472–481. Morgan Kaufmann, 1995.

[140] David V. Pynadath and Michael P. Wellman. Generalized queries on probabilistic context-free grammars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):65–77, jan 1998. ISSN 0162-8828.

[141] David V. Pynadath and Michael P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 507–514, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9.

[142] Xinzhou Qin and Wenke Lee. Attack plan recognition and prediction using causal networks. *Computer Security Applications Conference*, 0:370–379, 2004. ISSN 1063-9527.

[143] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. ISSN 0018-9219.

[144] Sindhu Raghavan and Raymond Mooney. Bayesian abductive logic programs. In *Proceedings of the AAAI-10 Workshop on Statistical Relational AI (Star-AI 10)*, pages 82–87, Atlanta, GA, July 2010.

[145] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1778–1783, 2009.

[146] Miquel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)*, 2010.

[147] Miquel Ramírez and Hector Geffner. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In Toby Walsh, editor, *International Joint Conference on Artificial Intelligence*, pages 2009–2014. IJCAI/AAAI, 2011. ISBN 978-1-57735-516-8.

[148] Silvia Richter, Malte Helmert, and Matthias Westphal. Landmarks revisited. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI)*, pages 975–982, 2008.

[149] Jussi Rintanen. An iterative algorithm for synthesizing invariants. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI)*, pages 806–811, 2000.

[150] Charles F. Schmidt, N. S. Sridharan, and J. L. Goodson. The plan recognition problem: an intersection of psychology and AI. *Journal of Artificial Intelligence Research*, 11(1,2):45–83, 1978.

[151] Mark Steedman. *The syntactic process*. MIT Press, Cambridge, MA, USA, 2000. ISBN 0-262-19420-1.

[152] Ruben Strenzke and Axel Schulte. Mixed-initiative multi-uav mission planning by merging human and machine cognitive skills. In *Proceedings of the Ninth International Conference on Engineering Psychology and Cognitive Ergonomics*, EPCE'11, pages 608–617, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21740-1.

[153] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262193981.

[154] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *In Pervasive*, pages 158–175, 2004.

[155] Dan Tecuci and Bruce Porter. A generic memory module for events. Key West, FL, 2007. Proceedings to the 20th Florida Artificial Intelligence Research Society Conference (FLAIRS).

[156] Edward P. K. Tsang. *Foundations of constraint satisfaction*. Computation in cognitive science. Academic Press, 1993. ISBN 978-0-12-701610-8.

[157] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979. ISBN 0-408-70929-4.

[158] Marc Vilain. Deduction as parsing: Tractable classification in the KL-ONE framework. In *Proceedings of the Ninth National Conference on Artificial*

Intelligence, volume 1 of *AAAI'91*, pages 464–470. AAAI Press, 1991. ISBN 0-262-51059-6.

[159] Mark Vilain. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI)*, pages 190–197, 1990.

[160] Daniel Weld and Oren Etzioni. The first law of robotics (a call to arms). In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1042–1047, Seattle, WA, 1994.

[161] Matthias Westphal and Silvia Richter. The LAMA planner. using landmark counting in heuristic search, 2008. Short paper for IPC 2008.

[162] Allen Y Yang, Roozbeh Jafari, S Shankar Sastry, and Ruzena Bajcsy. Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments*, 1(2): 103–115, 2009.

[163] Sung Wook Yoon, Alan Fern, and Robert Givan. FF-replan: A baseline for probabilistic planning. In Boddy et al. [24], pages 352–359. ISBN 978-1-57735-344-7.

[164] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22:179–214, April 2004. ISSN 1046-8188.