Hybrid STAN: Identifying and Managing Combinatorial Optimisation Sub-problems in Planning

Maria Fox and Derek Long
University of Durham, UK
Maria.Fox@dur.ac.uk, D.P.Long@dur.ac.uk

Abstract

It is well-known that planning is hard but it is less well-known how to approach the hard parts of a problem instance effectively. Using static domain analysis techniques we can identify and abstract certain combinatorial sub-problems from a planning instance, and deploy specialised technology to solve these sub-problems in a way that is integrated with the broader planning activities. We have developed a hybrid planning system (STAN4) which brings together alternative planning strategies and specialised algorithms and selects between them according to the structure of the planning domain. STAN4 participated successfully in the AIPS-2000 planning competition. We describe how sub-problem abstraction is done, with particular reference to route-planning abstraction, and present some of the competition data to demonstrate the potential power of the hybrid approach.

1 Introduction

The knowledge-sparse, or domain-independent, planning community is often criticised for its obsession with toy problems and the inability of its technology to scale to address realistic problems. Planners using weak heuristics, which attempt to guide search using general principles and without recourse to domain knowledge, cannot compete, in a given domain, against a planner tailored to perform well in that domain.

On the other hand, tailoring a planner to a particular domain requires considerable effort on the part of a domain expert. This effort is generally not reusable because a different domain requires a whole new body of expertise to be captured and it is not clear what (if any) general principles can be extracted from any single such effort to facilitate the next. The philosophy underlying our work on domain analysis is that knowledge-sparse planning can only be proposed as realistic general planning technology if it is supplemented by sophisticated domain analyses capable both of assisting a user in the development of correct domain descriptions and of identifying structure in a planning domain that can be effectively exploited to combat search.

In this paper we describe a way of decomposing planning problems to identify instances of NP-hard sub-problems, such as Travelling Salesman, that are most effectively solved by purpose-built technology. Knowledge-sparse, general, planning is unintelligent because it uses the same methods to solve all problems, whether they genuinely require planning or are in fact instances of well-known problems that are themselves the topic of substantive research. A more powerful approach is to allow such sub-problems to be abstracted out of the planning problem and solved using specialised technology. The different problem-solving strategies must then be integrated so that they can cooperate in the solution of the original problem.

We have been experimenting with using the automatic domain analysis techniques of TIM [Fox and Long, 1998] to recognise and isolate certain combinatorial sub-problems and to propose a way in which their solution, by specialised algorithms, might be integrated with a knowledge-sparse planner.

The work described in this paper has been successfully implemented in version 4 of the STAN system (STAN4) and has proved very promising. STAN4 competed in the AIPS-2000 planning competition where it excelled in problems involving route-planning sub-problems and certain resource allocation problems involving a restricted form of resource. The data sets from the competition are discussed in Section 5. In STAN4, TIM selects between a forward and backward planning strategy depending on characteristic features of the domain. The forward planning component, FORPLAN, is integrated with simplified specialist solvers for certain simple route-planning and resource allocation sub-problems. In Section 3 we describe the components of the hybrid architecture of STAN4 and explain the integration of these components.

2 Recognising Generic Behaviours

TIM can identify a collection of *generic types* within a domain. Generic types [Long and Fox, 2000] are collections of types, characterised by specific kinds of be-

haviours, examples of which appear in many different planning domains. For example, domains often feature *transportation* behaviours since they often involve the movement of self-propelled objects between locations. TIM can identify *mobile* objects, even when they occur implicitly, the operations by which they move and the maps of connected locations on which they move. The analysis automatically determines whether the maps are static (for example, road networks) or dynamic (for example, corridors with lockable doors). The recognition of transportation features within a domain suggests the likelihood of route-planning sub-problems arising in problem instances.

TIM also recognises certain kinds of *resources* which restrict the use of particular actions in a domain. Their presence suggests that resource allocation sub-problems related to Multi-processor Scheduling or Bin Packing might arise. TIM is able to recognise the existence, in a domain, of finite renewable resources which can be consumed and released in units. STAN4 exploited this in the Freecell domain in the AIPS-2000 competition (see Figure 3) but we are not, yet, exploiting the presence of such resources in a robust way. This paper does not give details of resource handling in STAN4, but our recognition and handling of Multi-processor Scheduling problems is described in [Long and Fox, 2001].

TIM takes as input a standard, unannotated, STRIPS or ADL description of a domain and problem. Integration of specialised technology with the search strategy of a planner is easiest to achieve in a heuristic forward-search-based planner, so we have implemented a forward planner, FORPLAN, using a simple best-first search strategy. FORPLAN uses a heuristic evaluation function, based on solving the relaxed planning problem, similar to the approach taken by HSP [Bonet *et al.*, 1997] and Hoffmann's FF [Hoffmann, 2000]. Like FF, FORPLAN uses a relaxed version of GraphPlan to compute the relaxed plan estimate. The difference between FORPLAN and FF is that the relaxed plan is constructed for the *abstracted* planning problem - that is, the part of the planning problem that remains when operators and preconditions relating to the identified sub-problem have been removed. This gives us only part of the heuristic estimate. The heuristic estimate is then improved by estimating the cost of solving the removed sub-problem. This two-part process can result in much better estimates than those produced by FF. FORPLAN has no effective heuristic control, except when path-planning or resource allocation sub-problems can be abstracted, so that it is almost useless as a general planner.

3 The Hybrid Planner Architecture

Because FORPLAN is not effective as a general planner we cannot rely on it for solving problems that do not have the sub-problem characteristics described above. We therefore constructed a hybrid system for entry into the AIPS-2000 competition. Our intention, in the competition, was to demonstrate an effective means of abstracting sub-problems from planning instances. In order to be able to compete realistically we needed to be able to report results for problems that did not have these features. We therefore added STAN version 3 as an alternative planning strategy, yielding a hybrid of two planning strategies and two specialist solvers (one for solving route-planning sub-problems and one for solving resource-allocation sub-problems). TIM operates as an interface to the hybrid, selecting between its components according to the structure of the domain. A high-level view of STAN4 is presented in Figure 1.

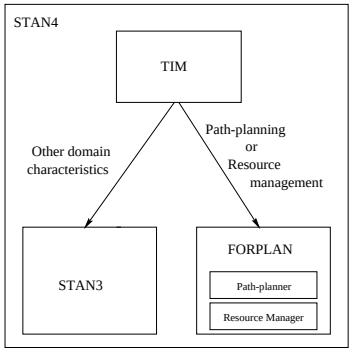

Figure 1: The architecture of the STAN4 hybrid system.

We now describe the processes by which route-planning sub-problems, once identified by analysis of a domain description, are abstracted and their solution, by specialised algorithms, integrated with FORPLAN. The processes by which resource-allocation sub-problems are handled are similar, but we do not describe them in detail here.

TIM first analyses the domain and problem instance to identify whether mobile objects exist, and if so whether a decomposable transportation problem can be found. TIM identifies the mobile objects and the locatedness predicate (also referred to as the *atrel*) they use (for example: *at*). The locatedness predicate is important for identifying actions which rely on changes in the locations of mobile objects. If an appropriate form of mobile is found TIM invokes FORPLAN together with the route-planning sub-solver. STAN3 is invoked in all domains in which neither route-planning nor resource allocation sub-problems can be found.

The presence of STAN3 means that TIM fails safe when it fails to identify a key sub-problem. At the moment this happens quite often because we are working with some simplifying assumptions about the structure of locatedness conditions and maps, and of resources, but we are gradually increasing the range of recognizable sub-problems.

If FORPLAN is invoked the domain must be modified, to abstract out the route-planning, or resource-allocation, sub-problem, before planning begins.

We intended STAN4 to demonstrate that a hybrid planning system, in which alternative planning strategies

are chosen automatically, depending on properties of the problem domain, is feasible. Our specialised algorithms for handling the abstracted sub-problems in STAN4 demonstrate that special-purpose approaches can be integrated successfully into the architecture of a planner. We make no great claims for the specific algorithms we used. For example, as described below, we use a simple nearest-neighbour heuristic to estimate the cost of solving a given Travelling Salesman instance when route-planning is abstracted. This results in poor estimates in some domains, for example when there are additional constraints present on the order in which locations can be visited. The nearest-neighbour heuristic was a first attempt at estimating the incurred costs - we are currently investigating alternative cost measures.

Although the GraphPlan-based strategy of STAN3 is now somewhat dated, its inclusion means that STAN4 can solve (at least some instances of) problems that cannot be effectively tackled by FORPLAN. STAN3 has a number of useful features, including symmetry handling [Fox and Long, 1999] and other search control mechanisms, which can still give it the advantage in problems that we cannot yet decompose. It also allows STAN4 to tackle instances of problems involving irreversible actions. Because FORPLAN does not backtrack over its choices it cannot reliably handle such domains.

Despite the valuable role played by STAN3 in the current hybrid it is invoked for largely negative reasons. We will, in the longer term, be interested in combining planning strategies for which stronger positive arguments can be made. We see the hybrid architecture as combining not just alternative planning strategies but collections of problem-solving strategies and specialised sub-solvers. There are a number of planners that combine alternative search strategies which are selected after the preferred strategy has been tried (for some predetermined period of time) and has failed (eg: Blackbox, FF, MIPS). Such systems are often referred to as hybrid, but the approach taken in these systems is different from the approach taken in STAN4. In STAN4 a particular planning or problem-solving strategy is not selected because of the failure of a preferred strategy but because of the suitability of the selected strategy to the perceived structure of the problem domain. The role of the domain analysis machinery of TIM in selecting between strategies is the key to the success of our hybrid planning approach.

The data presented in Section 5 shows the performance obtained in the AIPS-2000 planning competition on domains involving route-planning and resource-handling sub-problems. These domains were: Logistics and the STRIPS version of the elevator domain (route-planning) and Freecell (resource-allocation).

4 Sub-problem abstraction

In order to show how sub-problem abstraction is achieved in STAN4 we now describe in detail the process by which route-planning sub-problems are abstracted.

Having found that there are mobile objects in the domain TIM determines whether the problem of planning their movement between locations can be safely delegated to a sub-system. If the shortest distance to be travelled by an object moving from one location to another can be ascertained by looking at the map that the object moves on, then path-planning for that object can be devolved to a shortest path algorithm. If not (if the object can temporarily vacate the map) then a shortest path algorithm cannot be guaranteed to find the best path between two points. The object may be able to reappear on its map at a location different from the one it left, and this *flying* behaviour might give access to shorter paths than are visible on the map alone. If the object must always re-enter the map at the same location as the one it left then the shortest path between two points *is* guaranteed to be visible on the map so this restricted form of flying, which we call *hovering*, does not present a problem for route-planning abstraction.

We have experimented with a number of domains in which mobile objects having this hovering property arise. It sounds like a rather esoteric property, but it actually arises naturally. For example – in a simple *lamp-post maintenance* domain maintenance engineers move on a map defined by the locations of the lamp-posts, but they temporarily leave the map when engaged in a maintenance task (during these periods they "hover" above the map in raised maintenance vehicles) and then return to the same map location on completion of the task. In these domains abstraction of the route-planning sub-problem causes no difficulties because there is no shorter way to get between lamp-posts than to traverse the map. On the other hand, when flying is detected we do not attempt to abstract the problem of route-planning because it interacts in too complex a way with the rest of the planning problem.

4.1 Technical details

To achieve the abstraction of route-planning, once TIM has identified an appropriate mobile type, STAN4 associates with each mobile object a data structure which records the current location of the object. It also identifies each operator, other than the move operation of the mobile itself, the preconditions of which require a mobile of this type to be located at a particular location in order for the action to be executed. Once found, these preconditions are removed from the operators in which they appear, but each operator is then equipped with an additional data value identifying where the mobile must be in order to satisfy the abstracted precondition. In other words, the precondition is transformed from a standard proposition into a specialised representation with equivalent meaning, but allowing specialised treatment. This specialised representation (which we call a *mobile-tag*) provides the means of communication between the planner and a specialised sub-solver.

All move operations for the mobiles are then eliminated from the domain altogether. This results in an abstracted version of the domain containing the components of the original planning problem that the planner

will be required to solve. The problem is solved by the planner in this abstracted form. The abstracted problem is solved using FORPLAN, using a heuristic estimate of the value of a state based on the length of the relaxed plan between that state and the goal. The heuristic estimate is calculated by first constructing a relaxed plan with the abstracted operators, and calculating its length, and then adding to it an estimate for the lengths of the routes that would have to be traversed by any mobiles it uses. The latter calculation takes into account the cost of solution of the abstracted part of the problem.

The cost of traversing the routes that a plan entails is too expensive to compute with accuracy. The relaxed plan will show which locations each mobile is required to visit to satisfy the plan, with some ordering constraints imposed by the dependencies between the activities the mobile will be involved in at each location (loading must be carried out before unloading and so on). To calculate a shortest path that visits all these locations and respects these orderings is a variation on a Travelling Salesman problem, with multiple travellers and additional constraints. This problem is hard and cannot be solved repeatedly as part of the heuristic evaluation of a state. Instead, we produce an estimate of the cost by assuming that each mobile can visit each location in turn from the closest of the locations it has previously visited in the plan, respecting ordering constraints on the visits. Although this is an unsophisticated approach to tackling the Travelling Salesman problem, its integration with the planning process demonstrates the possibility of integrating more specialised technology. Despite its lack of sophistication it gives a better estimate of the cost of a state than a pure relaxed plan estimate, since relaxed plan estimates neglect the fact that a mobile cannot be in two places at the same time (the relaxed plan ignores delete conditions and it is these which express the fact that a mobile cannot be at two places at once).

Integration between the planner and the route-planner is required again when actions are selected for addition to the plan. Once an action is selected it is checked to determine whether it contains an abstracted locatedness precondition. If so, a path is proposed to move the mobile from its current location to the required destination (recorded within the mobile tag associated with action). We use the shortest path between the current location of the mobile (which is always known in a forward search) and the required location recorded in the mobile tag. At present this path is precomputed by TIM using Floyd's shortest paths algorithm on the map inferred from the initial state. This approach works well for static maps, where the shortest paths remain fixed, and in situations in which the movement consumes no additional resources. If the mobile does use resources during its movement, it might be that the shortest path is not the best, but instead a longer path which consumes fewer resources is to be preferred.

Finally, it is necessary to integrate the efforts of the planner and the route-planner to produce output in the form of a plan sequence. Once a route has been planned between the appropriate locations, STAN4 generates instantiations of the necessary move operators to produce a plan sequence corresponding to standard format for STRIPS plans. Below we present some of the preliminary results obtained using domains from the STRIPS subset of the AIPS2000 competition data set. In this collection of domains, Logistics and the MICONIC-10 lift domain both contain a path-planning sub-problem which TIM was able to identify and extract. Even using just our simple path-planning strategy we were able to obtain a significant performance advantage from exploiting path-planning abstraction.

5 Experimental Results

The data sets presented here were compiled by Fahiem Bacchus during the AIPS-2000 competition, held in Breckenridge, Colorado. In the graphs, the thick line plots the results of STAN4. Graphs showing time performance are log-scaled.

The graphs show how STAN4 performed on problems from the STRIPS data set involving either route-planning or resource allocation. The planners used for comparison are FF [Hoffmann, 2000], HSP-2 [Bonet and Geffner, 1997], TALplanner [Doherty and Kvarnstrom, 1999], SHOP [Nau *et al.*, 1999] and, occasionally, GRT [Refanidis and Vlahavas, 1999]. The problems used were Logistics, Freecell and the STRIPS version of the Miconic-10 elevator domain.

The competition comprised a fully-automated track and a hand-coded track in which planners were allowed to use hand-tailored domain knowledge. In the results presented here, STAN4, FF, GRT and HSP-2 are all fully-automated, whilst TALplanner and SHOP use hand-coded control knowledge.

STAN4 participated in the fully-automated track on the STRIPS problems. All planners able to handle the STRIPS version of PDDL competed in the STRIPS problems, including planners in the hand-coded track. However, despite the advantage of being supplied with hand-coded control knowledge, these planners did not consistently out-perform the fully automated planners. For example, STAN4 and FF were both faster than TALplanner and SHOP on the first Logistics data set (not shown) and produced at least as high quality plans.

From Figure 2 it can be observed that STAN4 took slightly longer than FF on the larger Logistics problems, but produced slightly better quality plans than any other planner, including those in the hand-coded track. As was emphasised earlier, the improvement in plan quality over FF derives from the fact that STAN4 uses a more informative heuristic than FF. STAN4 is using route-abstraction in this domain and achieves a small but consistent improvement in plan quality as a result.

The Freecell domain, Figure 3, was introduced specially for the competition and is a STRIPS formalisation of a solitaire card game released under Windows. Freecell has a resource-allocation sub-problem, because the free cells are a restricted, renewable and critical resource.

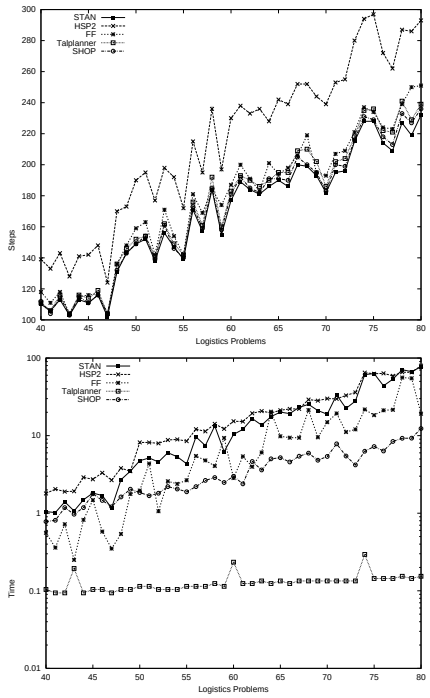

Figure 2: Quality of plans for, and time consumed to solve, Logistics problems 40-80

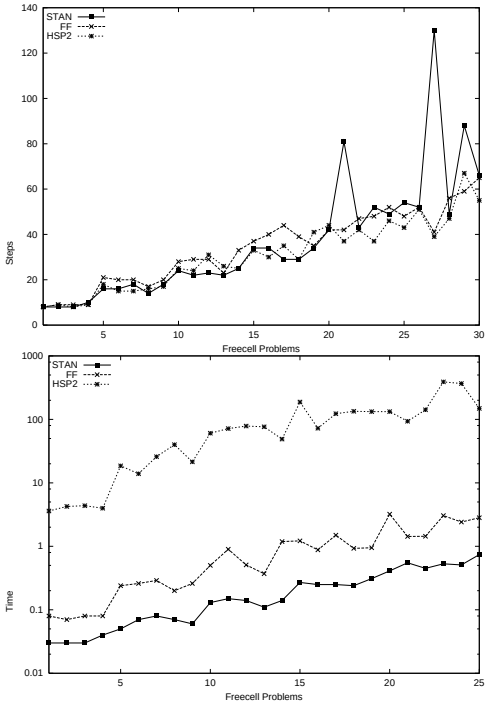

Figure 3: Quality of plans for, and time consumed to solve, Freecell problems.

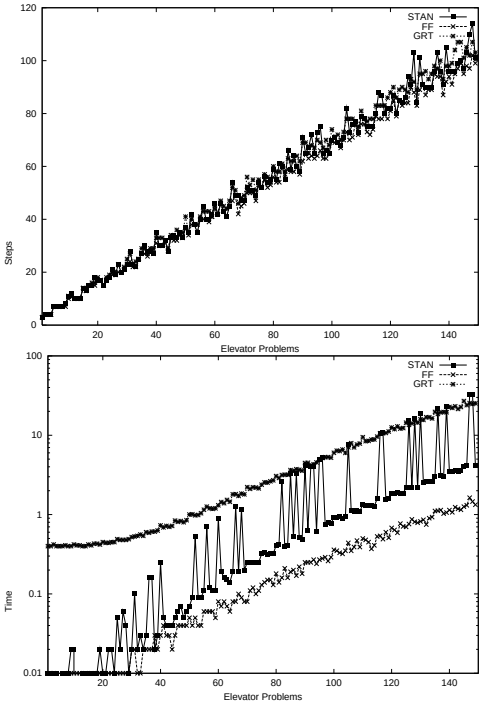

Figure 4: Quality of plans for, and time consumed to solve, STRIPS elevator problems.

To estimate how far a state is from the goal it is necessary to take into account the cost of ensuring that sufficient free cells are made available to meet the requirements of the abstracted relaxed plan. Our purpose-built technology for calculating this cost ensures that the consumption of resources does not exceed availability of those resources. If a plan entails over-consumption then the cost of sufficient release actions to redress the balance is added in to the estimate of its value. We have not yet succeeded in obtaining a robust way of accurately estimating these costs, and the performance of STAN4 is somewhat inconsistent as can be seen from the graph. Despite being fastest in all of the problems that it could solve, STAN4 missed several problems and was unable to solve any of the larger instances. Its plan quality was generally good, except for some anomalously long plans. More work is needed to adequately estimate the cost of distributing resources efficiently throughout a plan.

As Figure 4 demonstrates, STAN4 was the second fastest planner in the STRIPS elevator domain. Again, STAN4 is using route-planning abstraction, but produces slightly poorer quality plans at the the top end, than either FF or GRT, because of the coarseness of the nearest-neighbour heuristic used to solve the Travelling Salesman problem that arises for the elevator. This heuristic favours visiting all of the pick-up locations before any of the drop-off locations (the simplest way of respecting the ordering constraints in the plan). In fact, a subtler approach would be to allow the drop-off locations to be inter-mingled with the pick-up ones, provided

that a drop-off location is only selected next when the necessary people are on board. The nearest-neighbour heuristic tends to work less well whenever there are many objects to be transported (and many locations to be visited), and few carriers, as well as additional constraints (derived from the need to collect objects before delivering them) as in the elevator domain. The heuristic results in greater accuracy in Logistics because there are (typically) few packages to be transported by any one carrier. However, the nearest-neighbour heuristic was only ever intended to demonstrate that it is possible to integrate purpose-built machinery into the heuristic estimate, allowing the incurred cost of solving an abstracted problem to be taken into account in measuring the goodness of a state. We are currently investigating more sophisticated special-purpose algorithms.

The data presented here gives a clear indication of the potential value of sub-problem abstraction within a forward planning framework. Although FORPLAN is far from effective as a general planner, the exploitation of sub-problem abstraction makes a range of hard problems manageable and the generated solutions efficient.

6 Further Work

Although these foundations have produced promising results the framework we have used to achieve integration is somewhat unsophisticated and inflexible. TIM currently only recognises certain specific forms of mobile and very restricted forms of resource. As TIM fails safe when appropriate forms are not recognised this does not affect the completeness of STAN4. It does mean that STAN4 is often unable to exploit domain structure effectively and we are working on extending its repertoire.

STAN4 can only integrate with one specialised subsolver, even when there are two or more combinatorial sub-problems in a domain. At present STAN4 emphasises route-planning abstraction because we have made most progress in solving route-planning sub-problems effectively. An important development is to enable integration with more than one sub-solver. This will involve finding a way to communicate constraints between multiple sub-solvers and the planner.

Our "specialised technology" is currently very simplistic. An important refinement is to enable proper integration between the planner and the best available technology for solving combinatorial sub-problems where these arise. Our handling of resources in STAN4 is very restricted. We are working on the recognition of makespan subproblems, which are instances of Multiprocessor Scheduling, and their treatment using approximation algorithms for scheduling.

7 Conclusions

We have experimented, using STAN4, with the design of a hybrid planning system in which the choice of problem-solving strategy is made automatically following static analysis of the domain. Our current goals are to improve the integration between FORPLAN and the specialised solvers, allowing a more sophisticated profile of sub-problems to be managed, and to explore what advantages might be gained from integrating other planning strategies into the hybrid.

The key idea underlying our hybrid approach is that planning is not appropriate technology for solving all problems, and that resorting to generic search, or switching between a number of timed strategies, is not an effective way to address such problems. Instead we are interested in building up a collection of purpose-built strategies for combatting some of the most commonly occurring combinatorial sub-problems and making these available, together with techniques for recognising where these problems arise in planning domains. The decision about how to approach a given planning problem can then be made automatically, in a principled way, by deciding how to view the problem and deploying the most effective technology to solve it.

References

[Bonet and Geffner, 1997] B. Bonet and H. Geffner. Planning as heuristic search: new results. In *Proc. ECP*, 1997.

[Bonet *et al.*, 1997] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *AAAI*, 1997.

[Doherty and Kvarnstrom, 1999] P. Doherty and J. Kvarnstrom. Talplanner: An empirical inverstigation of a temporal logic-based forward chaining planner. In *Proceedings of 6th International Workshop on Temporal Representation and Reasoning*, 1999.

[Fox and Long, 1998] M. Fox and D. Long. The automatic inference of state invariants in TIM. *JAIR*, 9, 1998.

[Fox and Long, 1999] M. Fox and D. Long. The detection and exploitation of symmetry in planning problems. In *Proc. IJCAI*, 1999.

[Hoffmann, 2000] J. Hoffmann. A heuristic for domain-independent planning and its use in an enforced hill-climbing algorithm. Technical report, Albert-Ludwigs University, Freiburg, Germany, 2000.

[Long and Fox, 2000] D. Long and M. Fox. Automatic synthesis and use of generic types in planning. In *AIPS*, 2000.

[Long and Fox, 2001] D. Long and M. Fox. Multiprocessor scheduling problems in planning. Technical report, Department of Computer Science, University of Durham, UK, 2001.

[Nau *et al.*, 1999] D. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. SHOP: Simple hierarchical orederd planner. In *Proc. IJCAI*, 1999.

[Refanidis and Vlahavas, 1999] I. Refanidis and I. Vlahavas. GRT: A domain independent heuristic for STRIPS worlds based on greedy regression tables. In *Proc. ECP*, 1999.