

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

Maria Fox
University of Durham, UK
maria.fox@dur.ac.uk

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

November 13, 2000

Abstract

It is well-known that planning is a hard combinatorial problem but it is less well-known how to approach the hard parts of a problem instance effectively. Generic search is not always the most appropriate problem-solving tool, but planners do not generally have a wide repertoire of alternative strategies to apply to combinatorial sub-problems within planning domains. Some such (NP-hard) sub-problems occur very commonly – for example, *Travelling Salesman*-like sub-problems arise when a planning problem involves accomplishing tasks at different locations in contexts where the ordering in which these locations are visited affects the efficiency of the plan. *Multiprocessor Scheduling*-like problems occur when there are limited resources and tasks requiring the use of these resources. Again, the way in which restricted resources are allocated between processors can affect plan quality.

Using static domain analysis techniques we have been able to identify certain combinatorial sub-problems, within planning problems, making it possible to abstract these sub-problems from the overall goals of the planner and deploy specialised technology to handle them in a way integrated with the broader planning activities. We have experimented with the development of a hybrid planning system which brings together a range of planning strategies and specialised algorithms and selects between them according to the underlying properties of the planning domain. The hybrid planner constitutes version 4 of STAN, and it 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 [17]. Weak heuristics, which attempt to guide search using general principles and without recourse to domain knowledge, are not powerful enough to enable knowledge-sparse planning to 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, or making explicit expert knowledge that a planner might be able to exploit to avoid parts of the search space, requires considerable effort on the part of a domain expert. This effort is, in the main, 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. For planning technology to be attractive to a user there need to be ways of short-circuiting the domain

description process so that the domain designer is not required to specify what can be inferred from information already provided. 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 not best tackled by a general search strategy but are most effectively solved by purpose-built technology. Knowledge-sparse, general, planning is rather a blunt instrument for problem-solving 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 challenge then is to integrate the different problem-solving strategies 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[6] 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, our automatic domain-analysis system, 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. Full details of the abstraction process, its integration with planning and its limitations, can be found in [8].

2 Recognising Generic Behaviours

In our paper [12] we describe the process by which our automatic domain analysis tool, TIM [6], can identify a collection of *generic types* within a domain. Generic types are collections of types, characterised by specific kinds of behaviours, 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 or portable objects between locations. In the context of recognising transportation domains 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 *drivers* (if appropriate) on which their mobility depends, any objects they can carry, together with their associated *loading* and *unloading* operations. The analysis automatically determines whether the maps on which the mobiles move 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 planning problems within the domain.

Work is in progress, with promising initial results, allowing the recognition of certain kinds of

resources which restrict the use of particular actions in a domain. The presence of these features suggest that processor and resource allocation sub-problems might arise and might be related to combinatorial sub-problems such as Multi-processor Scheduling or Bin Packing. 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 to interesting effect in the Freecell domain in the AIPS-2000 competition (see Figure 6) but we have been unable, so far, to exploit the presence of such resources in a robust way. This paper does not, therefore, give details of our handling of these resources.

We are also working on the recognition of generic types indicative of other fundamental behaviours. For example, we have identified *construction* and *flow* as examples of generic behaviours associated with their own generic types and sub-problems. Construction problems are commonly associated with iterative behaviour, suggesting the presence in the domain of types with inductive structure (analogous to lists and trees) associated with well-defined inductive operations. Recognition of these features supports an abstract level of reasoning about the domain.

The analysis performed by TIM takes as input a standard STRIPS or ADL description of a domain and problem. No annotations are required to identify special behaviours and the analysis is entirely independent of the syntax used to describe the objects and operations. In addition, the analysis can identify generic behaviours in objects which do not obviously fall into the categories suggested by the names of the generic types. This allows the exploitation of specialist problem-solving technology in situations in which a human expert might not even recognise the presence of the appropriate generic type structure.

We have so far experimented most successfully with the recognition and abstraction of route-planning sub-problems. Integration of specialised route-planning 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 [3] and Hoffmann's FF [9]. Like FF, FORPLAN uses a relaxed version of GraphPlan [1] 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. As is described in the following sections, the heuristic estimate is then improved using the calculations performed to estimate 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 show that certain combinatorial sub-problems can be abstracted out of the planning process and to demonstrate a means of achieving this abstraction. 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 [11] version 3 as an alternative planning strategy, yielding a hybrid of two planning

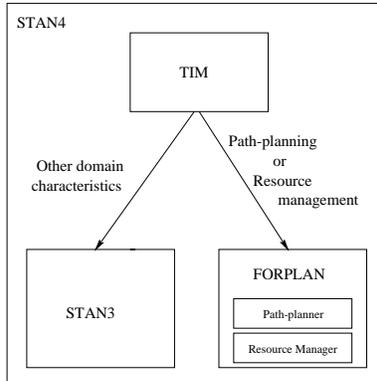


Figure 1: The architecture of the hybrid system showing how TIM selects between FORPLAN and STAN depending on the outcome of domain analysis.

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 components of the hybrid according to the structure of the domain. A high-level view of the architecture of STAN4 is presented in Figure 1.

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 the FORPLAN planning process.

STAN4 relies upon TIM for selection between the two available planning strategies. TIM first analyses the domain and problem instance. The objective of the analysis is 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. TIM then invokes FORPLAN or STAN3. FORPLAN is invoked if the domain is identified as having route-planning or resource-allocation sub-problems. STAN3 is invoked in all other domains. There are some positive reasons for invoking STAN3, but these have not yet been fully developed. For example, TIM attempts to identify whether the relaxed plan estimate is likely to be a good estimate of the distance between states, by measuring the extent to which actions impact on states. If a domain contains high impact actions then the relaxed plan estimates are likely to be poor. Our attempts to measure impact are currently too coarse to be robust, but we believe this is an important indicator of the suitability of forward search.

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, but we are gradually increasing the range of recognizable route-planning 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. A high level description of the algorithm is given in Figure 3.

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 (we discuss this issue in Section 5, with reference to the MICONIC elevator domain). The nearest-neighbour heuristic was a first approximation towards a way of estimating the incurred costs - we intend to replace this heuristic with a more appropriate cost measure and are currently investigating alternatives.

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 [7] 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. It is important to emphasise that 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. Our handling of resource allocation sub-problems is described in [13].

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. The issue is whether 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. If it can, 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

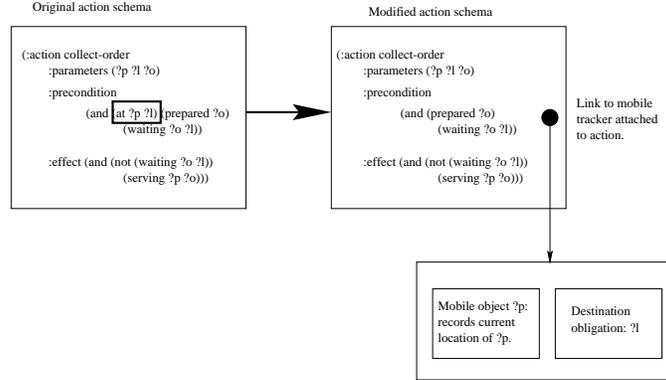


Figure 2: An operator schema being modified to remove preconditions concerning the mobile locat- edness predicate and replace them with a specialised representation storing the required location of the mobile. The rectangular box on the bottom is a *mobile tag*.

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

Preprocessing:

- (1) TIM identifies the mobile objects, their move operators and their atrels.
- (2) STAN4 abstracts the move operators out of the domain.
- (3) During instantiation of actions STAN4 abstracts out all the preconditions of other operators that are atrels.
- (4) STAN4 replaces each abstracted precondition with a tag saying where the mobile must be when this action is executed.

Planning:

- (1) FORPLAN estimates the distance from current to goal state using the abstracted domain.
- (2) To arrive at final estimate FORPLAN sums abstracted length and TSP route length obtained using NN heuristic.
- (3) When an action (in the abstracted domain) with a tag is selected, Floyd's is used to compute the shortest path from mobile current location to tag location.

Reporting a plan:

When the plan is output move sequences are generated to traverse the routes recorded in the plan.

Figure 3: Key steps of the STAN4 algorithm, taken when TIM recognises the domain as having mobile objects for which route-planning can be abstracted.

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. Clearly this cannot be solved efficiently in general and, still less, be solved repeatedly in the context of heuristic search, 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 [5] 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.

If the map is dynamic, additional goals will be introduced if the route requires new conditions to be satisfied in order for it to be opened along the way. These situations can be identified automatically, but it is harder to construct simple solutions to deal with them. An important focus of our current work is to improve both the sophistication of the cases that the route-planning can deal with and the integration of the route-planning with the planning to solve other problems.

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. Figures 1 and 2 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, during the fifth International Conference on AI Planning and Scheduling. In the graphs, the thick line is the line plotting the results of STAN4. Graphs showing time performance are log-scaled.

The graphs show how STAN4 performed, in comparison with a diverse range of the best-performing planners in the competition, on problems from the STRIPS data set involving either route-planning or resource allocation. The planners used for comparison are FF [9], HSP-2 [2], TALplanner [4], SHOP [15] and, occasionally, GRT [16]. The problems used were Logistics, Freecell and the STRIPS version of the Miconic-10 elevator domain. The Logistics domain was presented in two sets of problems. The problems increased in difficulty and the second set comprised larger (and hence harder) problems than the first.

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. HSP-2 is a refinement of HSP, the first forward planner to demonstrate the potential of using a relaxed plan heuristic. HSP (and HSP-2) assume independence of facts in the calculation of relaxed plan length, sometimes resulting in very pessimistic heuristic estimates.

STAN4 participated in the fully-automated track but could only handle the STRIPS problems. All planners able to handle the STRIPS version of PDDL [14] competed in the STRIPS problems, including planners in the hand-coded track. However, despite the advantage of being supplied with

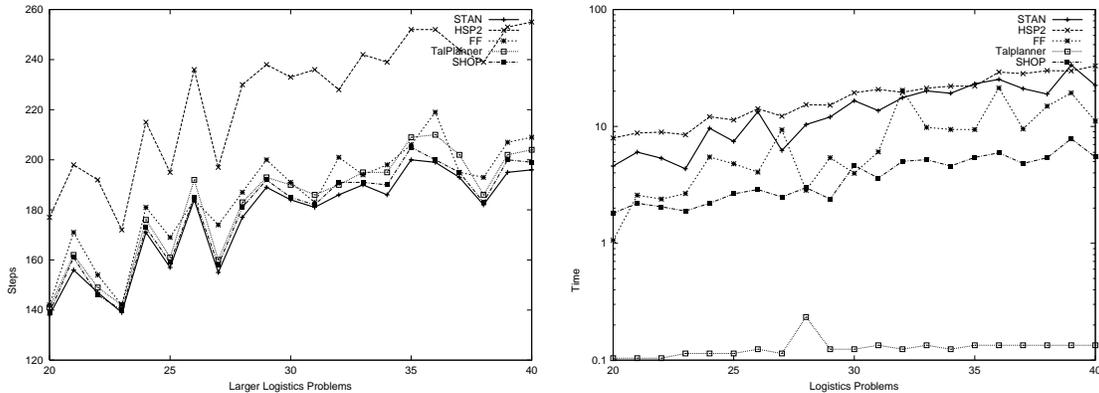


Figure 4: Quality of plans for, and time consumed to solve, problems 20-40 in the second Logistics problem set. The data compares three fully automated planners, STAN4, HSP-2 and FF, with two hand-tailored planners. STAN4 is producing the best quality plans because of its improved heuristic estimate.

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 dataset and produced at least as high quality plans (Figure 5).

From Figures 4 and 5 it can be observed that STAN4 took slightly longer than FF on the Logistics problems, but produced slightly better quality plans. As was emphasised earlier, the improvement in plan quality derives from the fact that STAN4 uses a more informative heuristic than FF, to guide its search. STAN4 is using route-abstraction in this domain and achieves, on average, a ten per-cent improvement in plan quality as a result.

Figure 6 shows how STAN4 performed in the Freecell domain. This domain was introduced specially for the competition, by Fahiem Bacchus, and is a STRIPS formalisation of a solitaire card game released under Windows. Freecell has a form of resource-allocation occurring as a sub-problem, because the free cells are a restricted, renewable and critical resource. In order 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 the balance between consumption and release deteriorates, so that there is the threat of over-consumption, then the cost of sufficient release actions to redress the balance is added in to the estimate. 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 Figure 6. 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 at the top end of the range that it could tackle. More work is needed to understand the nature of resource problems and to adequately estimate the cost of distributing resources efficiently throughout a plan.

Figure 7 demonstrates the performance of STAN4 in the STRIPS Miconic domain. STAN4 had a bug when the competition data was compiled, causing it to take a lot of time to solve some of the problems. This was later fixed so that these problems were brought in line with the rest of the data set. Again, STAN4 is using route-planning abstraction, but produces slightly poorer quality plans

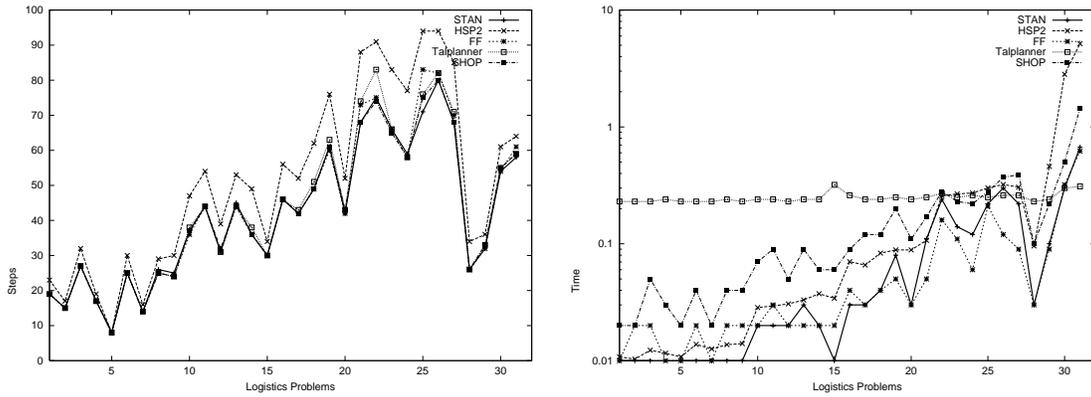


Figure 5: Quality of plans for, and time consumed to solve, problems 0-30 in the first Logistics problem set. FF and STAN are fastest - TALplanner is slower than the fully automated planners on this problem set, and immune to variations in the problems. It overtakes the other planners at around problem 30.

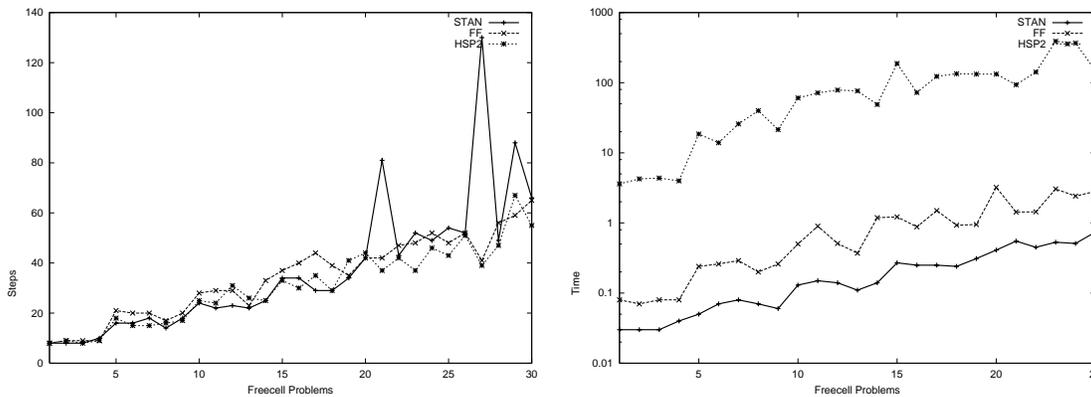


Figure 6: Quality of plans for, and time consumed to solve, Freecell problems. STAN4 is fastest on all of the problems it could solve and produces the best quality plans whilst its performance is stable (up to problem 20). STAN4 has abstracted out the handling of the allocation of the free cells and is using an estimate of the cost of balancing the number of used cells with the number of required cells to improve the relaxed plan estimate. Our handling of resource-allocation problems is unsophisticated at the moment, accounting for the variable performance.

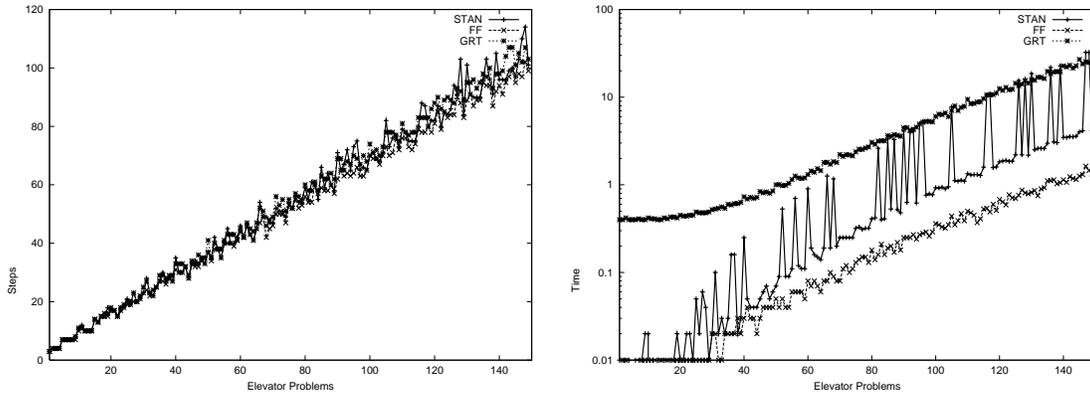


Figure 7: Quality of plans for, and time consumed to solve, STRIPS elevator problems. STAN4 produces slightly poorer quality plans, in some instances, than FF and GRT, and takes twice as long as necessary for about half of the problems because of a bug (now fixed). STAN4 is the middle set of data in the right hand graph - the density of the points makes it hard to distinguish the thick line! STAN4 is using route-planning abstraction. The failure of STAN4 to excel in this domain is due to the weakness of the nearest-neighbour heuristic, as described in the text.

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 be a problem 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. As explained in Section 3, 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. There are more sophisticated special-purpose algorithms that can be used and we are currently investigating these.

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.

The integration of FORPLAN with the sub-solvers involves constraint satisfaction, and it is reasonable to ask whether CSP technology might provide a more powerful framework for handling this reasoning within STAN4. However, CSP technology is generic and the problem of maintaining a set of consistent constraints is as hard as planning itself. Our strategy is to define sub-problem specific mechanisms for handling the integration of constraints sets with the planning process. Although our current mechanisms are unsophisticated in certain respects we believe that the use of inferred domain structure is critical in deploying appropriate constraint-handling technology to

make a planning problem more amenable to efficient solution.

6 Further Work

Although the foundations just described have produced promising results the framework we have used to achieve integration is somewhat unsophisticated and inflexible. The following weaknesses need to be addressed.

TIM only recognises that route-planning abstraction is possible if *all* mobiles in the domain are of an appropriate type to enable abstraction under our current assumptions. STAN4 therefore abstracts path-planning for all mobiles or none. An important refinement is to allow path-planning abstraction to be done for appropriate mobiles and not others.

STAN4 can only integrate with one specialised sub-solver, even when there are two or more combinatorial sub-problems evident in a domain. For example, if a domain involves route-planning and resource allocation STAN4 must choose just one of them to abstract. At present STAN4 chooses route-planning abstraction because we have made most progress in solving route-planning sub-problems effectively. An important refinement 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.

The sub-solver currently used to solve the Travelling Salesman sub-problem, during the heuristic estimate stage, is too simplistic and does not exploit state-of-the-art Travelling Salesman technology. An important refinement is to enable proper integration between the planner and the best available technology for solving combinatorial sub-problems where these arise. Some of the best solutions to Travelling Salesman are local search algorithms [10] which it would be interesting to integrate into the hybrid system.

7 Conclusions

We have experimented 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 hybrid system, STAN4, gave a promising performance in the AIPS-2000 competition, but it is currently restricted in terms of the kind of sub-problem integration that can be supported.

Our primary goal is to improve the integration between FORPLAN and the specialised solvers, allowing a more sophisticated profile of sub-problems to be managed. Our secondary goal is to explore what advantages might be gained from integrating other planning strategies into the hybrid and what positive reasons might be arrived at, through domain analysis, for selecting these strategies. 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 thrashing 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 optimisation problems and making these available to a planner, 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

- [1] A. Blum and M. Furst. Fast Planning through Plan-graph Analysis. In *IJCAI*, 1995.
- [2] B. Bonet and H. Geffner. Planning as heuristic search: new results. In *Proceedings of the European Conference on Planning (ECP)*, 1997.
- [3] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *AAAI*, 1997.
- [4] P. Doherty and J. Kvarnstrom. Talplanner: An empirical investigation of a temporal logic-based forward chaining planner. In *Proceedings of 6th International Workshop on Temporal Representation and Reasoning*, 1999.
- [5] R. W. Floyd. Algorithm 97: shortest path. *CACM*, 5(6), 1962.
- [6] M. Fox and D. Long. The automatic inference of state invariants in TIM. *JAIR*, 9, 1998.
- [7] M. Fox and D. Long. The detection and exploitation of symmetry in planning problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.
- [8] M. Fox and D. Long. The use of static analysis to identify and decouple sub-problems in planning. Technical report, Department of Computer Science, University of Durham, UK, 2000.
- [9] 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.
- [10] D. S. Johnson. Local optimisation and the Travelling Salesman Problem. In *Automata, Languages and Programming: Proceedings of 17th International Colloquium*, 1990.
- [11] D. Long and M. Fox. The efficient implementation of the plan-graph in STAN. *JAIR*, 10, 1999.
- [12] D. Long and M. Fox. Automatic synthesis and use of generic types in planning. In *International Conference on AI Planning and Scheduling*, 2000.
- [13] D. Long, M. Fox, L. Sebastia, and A. Coddington. An examination of resources in planning. Technical report, Department of Computer Science, University of Durham, UK, 2000.
- [14] D. McDermott. PDDL – the planning domain definition language. Technical report, Yale University, <http://www.cs.yale.edu/users/mcdermott.html>, 1998.
- [15] D. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. SHOP: Simple hierarchical ordered planner. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.
- [16] I. Refanidis and I. Vlahavas. GRT: A domain independent heuristic for STRIPS worlds based on greedy regression tables. In *Proceedings of the Fifth European Conference on Planning, Durham, UK*, 1999.
- [17] D. Wilkins and M. desJardins. A call for knowledge-based planning. In *AIPS workshop on Analyzing and Exploiting Domain Knowledge for Efficient Planning*, 2000.