# Hybrid STAN: Using Automatic Decomposition to Efficiently Manage Combinatorial Optimisation in Planning Problems

Maria Fox and Derek Long
University of Durham, UK
Maria.Fox@durham.ac.uk
d.p.long@durham.ac.uk

September 26, 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. Using static domain analysis techniques we have been able to identify certain combinatorial optimization 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. Our hybrid planner, STAN4, participated successfully in the AIPS-2000 planning competition.

## 1 Introduction

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 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. We have been experimenting with using the automatic domain analysis techniques of TIM[4] 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. Full descrptions of the processes involved can be found in [6, 5].

The work described here 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 sub-problems involving a restricted form of resource.

## 2 Sub-problem Identification

The way in which TIM recognises the presence of combinatorial sub-problems in a planing domain builds upon its identification of *generic types*. Generic types [10] 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. The recognition of transportation features within a domain suggests the likelihood of route-planning sub-problems arising in planning problems within the domain.

Another generic behaviour is resource-allocation, one form of which is indicated by the existence of finite renewable resources which can be consumed and released in units. STAN4 exploited this kind of resource sub-problem to interesting effect in the Freecell domain in the competition.

Integration of specialised sub-solvers 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 [2] and Hoffmann's FF [7]. Like FF, FORPLAN uses a relaxed version of GraphPlan [1] to compute the relaxed plan estimate. However, unlike FF, FORPLAN uses a two-part process to compute this estimate. In FORPLAN 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 using the calculations performed to estimate the cost of solving the removed sub-problem. For example, in the case of route-planning abstraction, these calculations estimate the cost of traversing the routes between the locations that must be visited by any mobile objects used in the plan.

To calculate a shortest path that visits all of these locations and respects any additional orderings imposed by the plan is a variation on a Travelling Salesman problem and cannot be solved efficiently in general. Instead, we produce an estimate of the cost using a simple nearest-neighbour heuristic. This is an unsophisticated first attempt at integrating a solver for Travelling Salesman problems, and does not always perform well. For example, in the STRIPS elevator domain, plans contain additional ordering constaints requiring the elevator to visit a number of pickup locations before their corresponding drop-off locations. The simplest solution for the nearest-neighbour heuristic is to visit all the pick-up locations before any of the drop-off locations, resulting in poor estimates. When additional constraints of this kind are not present (as in Logistics) this heuristic gives a better estimate of the cost than a pure relaxed plan estimate can give, since relaxed plan estimates ignore the fact that a mobile cannot be in two places at once.

Once an action is selected, for addition to the plan, it is checked to determine whether it entails obligations that must be satisfied by the plan (for example, in route-abstraction, actions in the abstracted plan entail commitments on mobile objects to reach key locations). If so, a way of meeting these obligations is proposed. In route-abstraction STAN4 proposes the shortest path between the current location of the mobile (which is always known in a forward search) and the required location, which is recorded in the action as part of the abstraction process. At present this path is precomputed by TIM using Floyd's shortest paths algorithm [3] on the map inferred from the initial state. It is not always appropriate to assume that the shortest path is the best one to follow. TIM can recognise when there are additional domain features which would invalidate this assumption and prevents the use of route-abstraction in such cases.

Finally, it is necessary to integrate the efforts of the planner and the sub-solver to produce output in the form of a plan sequence. Once a sequence of steps has been planned STAN4 generates instantiations of the necessary operators to produce a plan sequence corresponding to standard format for STRIPS plans.

## 3 The Hybrid Architecture of STAN4

In order to be able to compete realistically in the competition we needed to be able to report results for problems that did not have generic features we could recognise. We therefore supplemented FORPLAN with STAN [9] version 3, yielding a hybrid, STAN4, of two planning strategies and two (currently simplistic) specialist solvers - one for solving route-planning sub-problems and one for solving resource-allocation sub-problems.

TIM selects between the two planing strategies according to the structure of the domain. If the domain contains a recognized route-planning or resource-allocation sub-problem FORPLAN is invoked. STAN3 is invoked in all other cases. If FORPLAN is invoked the domain is automatically modified, to abstract out the route-planning (or resource-allocation) sub-problem, before planning begins. This process, and its limitations, is described in [6]. A high-level view of the architecture of STAN4 is presented in Figure 1.

## 4 Further Work

Although STAN4 has produced some promising results the framework we have used to achieve integration is somewhat unsophisticated and inflexible. The following weaknesses are currently being 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

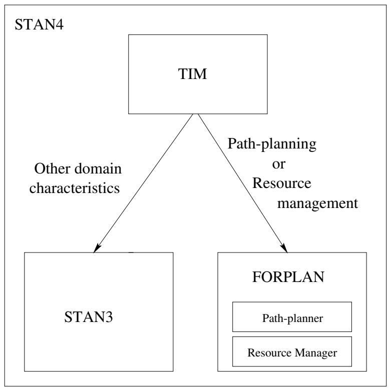
STAN4
TIM
Other domain characteristics
Path-planning or Resource management
STAN3
FORPLAN
Path-planner
Resource Manager

Figure 1: The architecture of the hybrid system showing how TIM selects between FORPLAN and STAN3 depending on the outcome of domain analysis.

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-solvers currently integrated with the hybrid are too simplistic and produce poor performance in some domains. 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 [8] which it would be interesting to integrate into the hybrid system.

## 5 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 integration with specialised solvers, allowing a more sophisticated profile of sub-problems to be managed. 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, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *AAAI*, 1997.

[3] R. W. Floyd. Algorithm 97: shortest path. *CACM*, 5(6), 1962.

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

[5] M. Fox and D. Long. Hybrid stan: Identifying and managing combinatorial optimisation sub-problems in planning. In *Proceedings of the 20th UK Workshop on Planning and Scheduling*, 2000.

[6] 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.

[7] 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.

[8] D. S. Johnson. Local optimisation and the Travelling Salesman Problem. In *Automata, Languages and Programming: Proceedings of 17th International Colloquium*, 1990.

[9] D. Long and M. Fox. The efficient implementation of the plan-graph in STAN. *JAIR*, 10, 1999.

[10] D. Long and M. Fox. Automatic synthesis and use of generic types in planning. In *International Conference on AI Planning and Scheduling*, 2000.