STAN4
A Hybrid Planning Strategy Based on Subproblem Abstraction

Maria Fox and Derek Long

■ Planning domains often feature subproblems such as route planning and resource handling. Using static domain analysis techniques, we have been able to identify certain commonly occurring subproblems within planning domains, making it possible to abstract these subproblems from the overall goals of the planner and deploy specialized technology to handle them in a way integrated with the broader planning activities. Using two such subsolvers our hybrid planner, STAN4, participated successfully in the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00) planning competition.

The philosophy underlying our work on domain analysis is that uninformed, knowledge-sparse planning is impractical for real application. Although such strategies can be impressive when applied to toy domains, they cannot address highly structured problem domains effectively. However, when knowledge-sparse approaches are supplemented by domain knowledge, they can perform impressively (Bacchus and Kabanza 2000) at the cost of an increased representation burden on the domain designer. We have been exploring the use of automatic domain analyses to identify structure in a planning domain that a planner can exploit to combat search.

In this article, we introduce a way of decomposing planning problems to identify instances of common subproblems. In many cases, high-performance approximation strategies exist for solving such problems, and it is inappropriate to address them using brute-force search. We have been experimenting with using the automatic domain analysis techniques of TIM (Fox and Long 2001a, 2001b, 1998; Long and Fox 2000a, 2000b) to recognize and isolate subproblems and integrate their solution, by means of specialized algorithms, with the search behavior of a knowledge-sparse planner. Full descriptions of the processes involved can be found in Fox and Long (2001a, 2001b).

A preliminary hybrid architecture was successfully implemented in version 4 of the STAN system (STAN4) and has proved very promising. STAN4 competed in the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00) planning competition where it excelled in problems involving route-planning subproblems and certain simple resource-allocation subproblems involving a restricted form of discrete, reusable resource. This article describes the key features of the competition version of STAN4.

The Architecture of STAN4

The way in which TIM recognizes the presence of combinatorial subproblems in a planning domain builds on its identification of generic types.

Generic types (Long and Fox 2001, 2000a) are collections of types, characterized by specific kinds of behaviors, examples of which appear in many different planning domains. For example, domains often feature transportation behaviors because 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 subproblems arising in planning problems within the domain.

Another generic behavior is resource allocation, one form of which is indicated by the existence of finite renewable resources that can be consumed and released in units. STAN4 exploited this kind of subproblem to interesting effect in

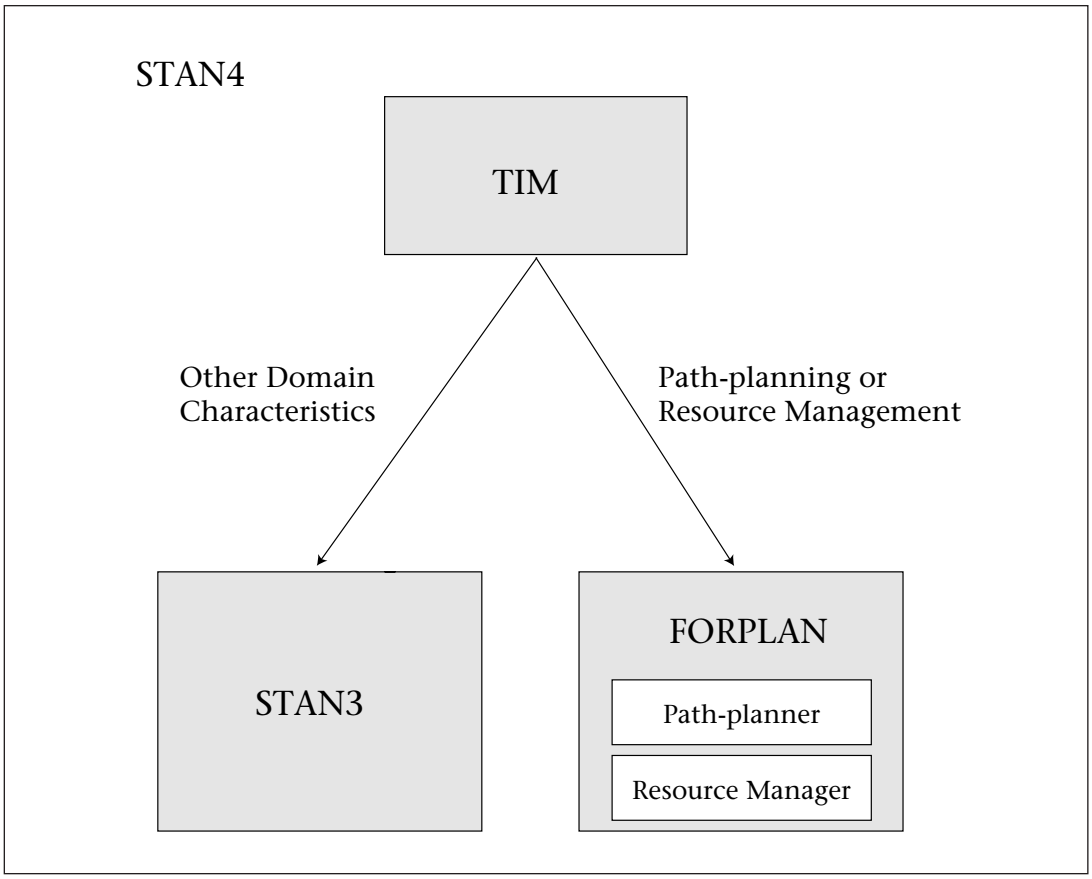

Figure 1. The Architecture of the Hybrid System Showing How TIM Selects between FORPLAN and STAN3 Depending on the Outcome of Domain Analysis.

the *FreeCell* domain in the competition.

The competition version of STAN4 consists of the integrated combination of TIM, two distinct planning strategies, and two specialized subsolvers. Given a problem, TIM is responsible for invoking an appropriate planner and a specialized strategy. Integration of the specialized technology with a planner is easiest to achieve in a heuristic forward-search–based planner because the states of all variables can be known at any point during the forward-search process. We therefore implemented a forward planner, FORPLAN, using a simple best-first search strategy. FORPLAN is only ever invoked when TIM recognizes that a planning domain contains a route-planning or resource-allocation subproblem. Before invoking FORPLAN, TIM modifies the domain description to abstract out the recognized subproblem. FORPLAN is then invoked on the simplified domain description.

FORPLAN uses a heuristic evaluation function, based on solving the relaxed planning problem, similar to the approach taken by HSP (Bonet, Loerincs, and Geffner 1997) and FF (Hoffmann 2000). Like FF, FORPLAN uses a relaxed version of GRAPHPLAN (Blum and Furst 1995) 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, giving only part of the heuristic estimate. The heuristic estimate is then improved using the calculations, performed by the appropriate specialized solver, to estimate the cost of solving the removed subproblem. 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 these locations and respects any additional orderings imposed by the plan is a variation on a traveling salesman problem. We produce an estimate of the cost using a simple nearest-neighbor heuristic, which is an unsophisticated first attempt at integrating a solver for a traveling salesman problem and does not always perform well. For example, in the STRIPS elevator domain, abstract plans require the ele-

vator to visit a number of pick-up locations before their corresponding drop-off locations. The simplest solution for the nearest-neighbor heuristic is to visit all the pick-up locations before any of the drop-off locations, resulting in poor estimates. Better estimates could be obtained by interleaving the visits, but the nearest-neighbor heuristic cannot exploit this possibility. When such interleaving is not needed (as in logistics), this heuristic gives a better estimate of the cost than a pure relaxed-plan estimate can give. It can be observed, from the competition data, that STAN4 produced the best-quality plans in all the logistics problems it could solve, whereas its performance in the elevator domain was variable.

Once an action is selected, 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 by the specialized algorithm. In the case of route abstraction, the subsolver 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. STAN4 uses Floyd's (1962) algorithm to compute this path in static maps.

To be able to compete realistically in the competition, we needed to be able to report results for problems that did not have either of the two generic features that TIM could recognize at this stage in its development. We therefore supplemented FORPLAN with STAN3 (Long and Fox 1999), a GRAPHPLAN-based planner exploiting various preprocessing techniques. We then integrated our route-planning subsolver, and a resource-allocation subsolver, with the FORPLAN strategy.

An important feature of STAN4 is that TIM fails safe when it is unable to detect an appropriate generic structure in a domain. If a domain features a more complex form of mobile than TIM can recognize, then STAN4 is unable to exploit the presence of this mobile, but it can still solve the problem using the uninformed planning techniques of STAN3. Thus, despite its specialized handling of domains when appropriate, STAN4 is a domain-independent system.

TIM selects between the two planning strategies according to the structure of the domain. If the domain contains a recognized subproblem, FORPLAN is invoked. STAN3 is invoked in all other cases. If FORPLAN is invoked, the domain is automatically modified to abstract out the subproblem before planning begins. This process is described in Fox and Long (2001a, 2001b). A high-level view of the architecture of STAN4 is presented in figure 1.

Further Work

Although the competition version of STAN4 produced some promising results, the framework we used to achieve integration was unsophisticated and inflexible. Our subsequent work has addressed the following points:

First, in the competition version of STAN4, TIM could only recognize simple mobile types. TIM is now able to recognize a more diverse range of mobile types and perform appropriate modifications to enable the route-planning subproblems to be abstracted and handled correctly. TIM can now identify mobiles that need drivers to provide their mobility and can solve the driver-allocation subproblem as part of route planning for these mobiles.

Second, the competition version of STAN4 featured a simplistic integration mechanism, which was based on attaching subproblem-specific information to domain constructs during the abstraction process, allowing FORPLAN to pass specific forms of information to the two subsolvers. Subsolvers could only pass back heuristic distance estimates. The support for two-way communication has now been improved and generalized into a proper interface between the planning and subsolving levels (Fox and Long 2001a).

Third, in the competition version, STAN4 only supported integration with one specialized subsolver at a time, even when a domain featured multiple subproblems. The uniform interface mentioned earlier supports a more powerful form of integration, allowing multiple subsolvers to communicate with the planner. This area is a focus of ongoing research.

Fourth, the subsolvers integrated with the competition version are simplistic and produce poor performance in some domains. The competition version of STAN4 was designed as a proof of concept, demonstrating that domain analysis could be used to select intelligently between alternative problem-solving strategies. The use of more powerful specialized techniques is now being considered.

The key idea underlying our hybrid approach is that uninformed search is not appropriate technology for solving all planning problems.

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 preliminary hybrid system, STAN4, gave a promising performance in the AIPS'00 compe-

tition but was restricted in terms of the power of integration that could be supported.

The competition data show that in domains in which TIM was able to identify exploitable subproblems, STAN4 gave a good performance, particularly in terms of plan quality. Its logistics plans were better even than those of TALPLANNER, which was using hand-coded control knowledge. In the *FreeCell* and elevator domains, STAN4 gave a varied performance, demonstrating that the identification of an exploitable subproblem, and integration of appropriate subsolvers, was still at a raw stage at the time of the competition.

Our primary goal since the competition has been to improve integration with specialized solvers, allowing a more sophisticated profile of subproblems to be managed. The key idea underlying our hybrid approach is that uninformed search is not appropriate technology for solving all planning problems. Instead, we are interested in building up a collection of purpose-built strategies for combatting some of the most commonly occurring subproblems and making these available to a planner, together with techniques for recognizing 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

Bacchus, F., and Kabanza, K. 2000. Using Temporal Logic to Express Search Control Knowledge for Planning. In *Artificial Intelligence, Volume 16,* 123–191. New York: Elsevier Science.

Blum, A., and Furst, M. 1995. Fast Planning through Planning Graph Analysis. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 1636–1642. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A Robust and Fast Action Selection Mechanism for Planning. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, 714–719. Menlo Park, Calif.: American Association for Artificial Intelligence.

Floyd, R. W. 1962. Algorithm 97: Shortest Path. *Communications of the ACM* 5(6):345.

Fox, M., and Long, D. 2001a. Extending the Recognition and Use of Mobility in Planning Domains. Technical report, Department of Computer Science, University of Durham.

Fox, M., and Long, D. 2001b. Hybrid STAN: Identifying and Managing Combinatorial Optimization Subproblems in Planning. Paper presented at the Seventeenth International Joint Conference on Artificial Intelligence, 4–11 August, Seattle, Washington.

Fox, M., and Long, D. 1998. The Automatic Inference of State Invariants in TIM. *Journal of AI Research* 9:367–421.

Hoffmann, J. 2000. A Heuristic for Domain-Independent Planning and Its Use in an Enforced Hill-Climbing Algorithm, Technical report, Institute for Computer Science, Albert-Ludwigs University.

Long, D., and Fox, M. 2001. Planning with Generic Types. Technical report, Department of Computer Science, University of Durham.

Long, D., and Fox, M. 2000a. Automatic Synthesis and Use of Generic Types in Planning. In *Proceedings of the Fifth International Conference on AI Planning and Scheduling,* 196–205. Menlo Park, Calif.: AAAI Press.

Long, D., and Fox, M. 2000b. Extracting Route Planning: First Steps in Automatic Problem Decomposition Workshop on Analyzing and Exploiting Domain Knowledge for Efficient Planning. Presented at the Fifth International Conference on Artificial Intelligence Planning and Scheduling, 14–17 April, Breckenridge, Colorado.

Long, D., and Fox, M. 1999. The Efficient Implementation of the Plan-Graph in STAN. *Journal of AI Research* 10:87–115.

Maria Fox is reader in computer science at the University of Durham, United Kingdom, where she has worked since 1995. She previously held a lectureship at University College London. Fox has been working in AI planning since obtaining her Ph.D. in 1989. In collaboration with Derek Long, she developed the STAN and TIM systems. Her main interests are in domain-independent planning and preplanning domain analysis techniques. Her e-mail address is maria.fox@durham.ac.uk.

Derek Long is a lecturer in computer science at Durham University, United Kingdom. He joined the department in 1995 after lecturing at University College London for six years. He obtained his doctorate at the Programming Research Group, Oxford University, in 1989. Since then, he has pursued interests in reasoning techniques in general and in planning in particular. He has worked in close collaboration with Maria Fox for the last 10 years. His e-mail address is d.p.long@durham.ac.uk.