Generic Types and their Use in Improving the Quality of Search Heuristics

Andrew Coles
Department of Computer and Information Sciences
University of Strathclyde
26 Richmond Street,
Glasgow, G1 1XH
email: andrew.coles@cis.strath.ac.uk

Abstract

This abstract discusses work looking into techniques for improving the quality of the search heuristics used to guide forward-chaining planning. The improvements in heuristic quality are made by performing a static analysis of the planning problem to identify commonly occurring 'generic types', and providing additional heuristic guidance based on their known properties. In doing so, the heuristic is tailored to the identified properties of the domain and can provide a more realistic heuristic value and refined relaxed plan. This can potentially lead to reduction in the time taken to find a plan, and the generation of shorter plans.

Introduction

Forward-chaining planning guided by a heuristic has proved to be an effective planning strategy in a range of planning domains. At recent international planning competitions, many of the participating planners followed this search approach; of particular note is FF (Hoffmann & Nebel 2001), which participated with great success in the 2002 and 2000 competitions. Work on HSP (Bonet & Geffner 2000) and Downward (Helmert 2004) has explored alternative heuristics. What all these planners share, however, is that the heuristic goal-distance estimate they provide is obtained from a 'relaxed' version of the original problem, i.e. one from which some constraints have been removed. The relaxation of the original problem in this manner is necessary to allow a heuristic value to be obtained in a reasonable time; however, it does reduce the accuracy with which the relaxed problem is able to model certain aspects of the original problem.

Using static analysis techniques, such as those performed by TIM (Long & Fox 2000), it is possible to identify 'generic types' of objects within planning problems: for instance, self-propelled mobile objects capable of moving from one location to another. These generic types form subproblems with known properties with which type-specific heuristics can be used: for instance, using the Floyd Walshall algorithm to calculate the cost of moving a mobile from one location to another. HybridSTAN (Fox & Long 2001), a forward-chaining heuristic planner, took the approach of isolating these known sub-problems when planning, removing all predicates pertaining to the location of mobiles from the domain. Once a solution plan was found, actions were inserted into the plan to move the mobile objects to the locations needed for the actions used.

The decomposition approach of HybridSTAN relies on being able to cleanly isolate the sub-problem, which is only possible if it is wholly described by the generic type. For example, if the move action for the mobile requires another condition to be satisfied (such as one defining whether a door is open between the two locations) then the subsolver cannot handle the additional constraints imposed. In these cases, is not possible to add the missing actions to the plan as required once the remainder of the problem has been solved, as it is no longer clear which actions are needed.

To this end, this work is concerned with investigating whether the static domain analysis used to discover subproblems can be used to improve the quality of the relaxation heuristic used, in this case the Relaxed Planning Graph heuristic, without relying on being able to solve the identified subproblems in isolation. By improving the heuristic, and the guidance it provides through state space, the aim is to reduce the time taken to find solution plans and to improve the quality of plans found.

Background

Generic Types

TIM is capable of identifying objects, or groups of objects, within planning problems as having a recognisable generic behaviour and thus being of a certain generic type. TIM first analyses planning problems to discover the 'property spaces' relating to each of the objects. From these, generic types are identified by looking for hand-coded patterns of transitions within the property spaces. Included in these generic types are mobiles and resources. Mobiles have a location property, the value of which is changed by the application of 'move' actions to move the mobile from one location value to another. The locations at which the mobile can be located are arranged into a map; directed edges exist in the map between pairs of locations where a feasible move action exists to move the mobile from the source location to the destination. At no point, either in the initial state or any sound, reachable, state is is it possible for a mobile to be located at more than one location.

Resources are a special case of mobiles, whose map consists of a series of linearly interconnected nodes. An edge can be drawn from a node A to a node B if there exists an action capable of moving the mobile denoting the resource level from A to B; an edge can be drawn from B to A if there exists an action capable of moving the resource level from B to A. Edges in one direction correspond to increasing the resource level; edges in the other direction correspond to decreasing the resource level.

Known generic types can sometimes arise in unexpected situations, where human intuition might not have expected them. Any object which has a predicate relating it to one of a series of other objects and a corresponding action schema which changes this assignment is identified as a mobile object.

Generic Types and the Relaxed Planning Graph Heuristic Landscape

The relaxed planning graph heuristic, as first used in FF, has proved to be a useful heuristic for guiding forward-chaining planning. The relaxation used as a basis for the heuristics is to ignore the delete lists (negative effects) of the domain actions; Graphplan (Blum & Furst 1995) is then used to solve this relaxed problem, although only a subset of the algorithm needs to be implemented as the planning graph does not contain mutexes due to the removal of delete effects.

When delete lists are ignored, once a fact has been established by an action, it is available for use as a precondition to all the subsequent actions in the plan. This has some interesting effects on how well the relaxed problem is able to model some aspects of known generic types within planning problems. When the move actions of mobile objects are invoked, the effects of the action normally establish two facts: the mobile is now located at the destination; and the mobile is no longer located at the source. Similarly, when action increasing or decreasing resource levels are invoked: the resource level is now that resulting from the action; and no longer holds the previous value. Ignoring the delete effects of move actions (or resource-level-altering actions), as done when forming the relaxed planning problem, removes the effects that establish that once a mobile has moved it is no longer at its previous location. Effectively, when executing a relaxed plan, mobiles are simultaneously available at all the locations they have ever been, and resources are available at all levels they have held.

When dealing with resources, this can have a substantial impact on how well the relaxed planning problem models the original: if a resource level is non-zero in the initial state from which a relaxed-plan is built, it is available at that non-zero level throughout. In FreeCell, for instance, if there is one free cell available in a given state, the relaxed plan to the goal from that state can make use of an effectively unlimited number of free cells. No action is able to reduce the number of free cells available by subsequent actions, as the delete effect that would establish that the free cell count is lowered when a card is placed in a free cell has been removed. This can lead, for instance, to relaxed plans which state that as many cards as necessary should be moved to a free cell and then the cards should be moved to the home cells in the correct order.

This over-optimism in the presence of a resource level of one by the relaxed planning graph heuristic has a profound effect on search: up to, and including the point, where there is still a non-zero resource level, as much of the resource as desired is available so a relaxed solution plan can be found. However, as soon as an action reduces a resource level to zero, the nature of the relaxed plan changes dramatically: if that resource is required then actions must be added to the relaxed solution plan to increase the resource level (assuming such resource-increasing actions are available). This sudden change in relaxed plan can lead to unforeseen dead-ends, or a sudden increase in relaxed plan length - both of which have a negative impact on search performance.

The multi-locatedness of mobiles under the ignore-delete-lists relaxation—that is, a mobile is available at all the locations it has ever been at thus far in the relaxed plan—can lead to some interesting relaxed plans being formed. Consider, for instance, a logistics problem in which a truck, beginning in location A, must collect a package from location E and deliver it to location A. The relaxed plan forwards from the initial state moves the truck from A to E (via B, C and D), loads the package into the truck and immediately unloads it at A: this 'teleportation' of the package from E to A, without the truck having to move back again, occurs because the fact that the truck is in location A was never deleted and, thus, the unload action placing the package at A is immediately applicable.

Relaxed Plan Refinement using Generic Types

Refining Relaxed Plans

In many cases, one can identify actions that are logically missing from relaxed plans that would need to be inserted in order to make the plan executable if delete lists were considered. Through analysis of the behaviour of known-typed objects in the plan, it is possible to suggest what some of the missing actions are, and produce a relaxed plan which is somewhat 'less relaxed' than it was previously.

When dealing with mobiles, if a precondition of one action demands that a mobile be in one location, and the precondition an action immediately following it demands that it be in another, then it is clear that actions to move the mobile from the former location to the latter would be necessary. As the map describing how the mobile can traverse between its locations is known, a path between all possible pairs of locations that may arise can be determined, in polynomial time, using the Floyd Walshall algorithm. The additional actions corresponding to the mobile moving along this path can be added to the relaxed plan, making it a closer analogue of a real solution plan, and increasing the heuristic cost by number of actions added.

The level of a resource is denoted by an assignment to a series of ranked objects. Actions which increase the level of the resource change the assignment denoting the resource level to a higher-ranked object; actions which decrease the level of the resource change the assignment to a lower-ranked object. By starting with the resource level in the state from which the relaxed plan was built, the cumulative resource-level effects of the actions in the relaxed plan can be monitored: resource-increasing actions move the current resource level one place higher up the rank; resource-decreasing actions move it one place lower. If at any point an action attempts to move the resource level to off the top of the rank or off the bottom of the rank, a decreasing or increasing action needs to be inserted as appropriate. Such actions are not available in all cases; if they are not, a penalty can be added the heuristic value returned (the plan length) to dissuade search from considering plans whose relaxed solutions appear to violate resource limits. This is similar to the adjusted cost heuristic used in Sapa (Do & Kambhampati 2003), but as TIM provides finite bounds on the resource levels it is possible to penalise resource flows through the relaxed plan that would take the resource level both below and above its bounded values.

The effect of relaxed plan extraction on plan refinement

The process to extract a relaxed plan from a GraphPlan planning graph is designed to be as efficient as possible, to reduce the overhead of heuristic evaluation. When choosing an achiever for each fact, the first achiever found when building the planning graph is used. The first achiever found, however, varies between states, and can lead to dramatically different relaxed plans being built, even if the plan lengths are similar.

When refining the relaxed plans built in the conventional manner, the penalty is heavily dependent on the first-achieving actions found; in this case, adding actions to the relaxed plan adds noise to the relaxed plan length, making it difficult to decide which states are the most likely to lead to a goal. In an attempt to address this problem, two alternative plan extraction approaches are being investigated:

- A stochastic approach, called several times in an attempt to minimise noise, in which one of the achievers for each fact is chosen at random, rather than the first one found;
- A guided approach, called once, which uses a heuristic to choose which successor to use.

These alternatives will lead to differing heuristic values being found; which may lead to improved performance and/or shorter plans.

Using Lookahead with Refined Plans

The heuristics discussed are invariably more expensive than the baseline, unrefined, relaxed planning graph heuristic. The 'less-relaxed' plans found are, however, closer to being solutions to the original planning problem than unrefined relaxed plans; suggesting that it would be beneficial to use more than just the plan length as a heuristic value to guide search.

YAHSP (Vidal 2004), a planner which competed at the 2004 international planning competition, uses a lookahead approach to generate an additional successor to each state. The additional successor state is formed by applying as many of the sequenced actions from the relaxed plan as possible. In YAHSP, in an attempt to satisfy some of the unsatisfied preconditions of the actions in the relaxed plan, an attempt is made to find one action that would add the unsatisfied precondition. Adding action sequences to satisfy preconditions is not, however, considered: if satisfying a precondition requires more than one action, lookahead terminates.

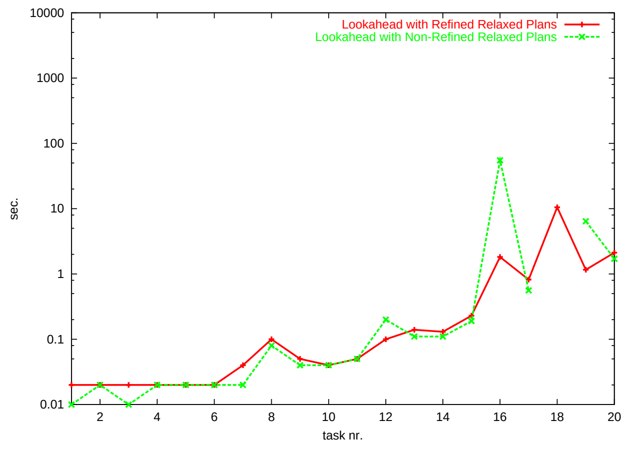

Figure 1: Time Taken To Solve Problems in the DriverLog Domain with Lookahead on Refined and Non-Refined Plans

Performing lookahead on the less-relaxed plan provided by the generic-type refinement, rather than the conventional relaxed plan, should allow more actions to be applicable in domains with recognised generic types. Within the refined plan, move action sequences to satisfy locatedness preconditions have been added; something which the lookahead procedure itself cannot do, as it only considers adding single actions to satisfy preconditions. The combination of these two techniques allows the low-cost of the lookahead procedure to be maintained, by it only considering adding single actions, whilst allowing action sequences to be inserted where these can be determined using the generic types analysis.

Using lookahead provides a further possibility: using the non-refined relaxed plan to provide a heuristic value; but performing lookahead over the refined plan. Such a configuration would have two benefits:

- lookahead can apply more actions than it would have done otherwise, as action sequences to achieve mobile locations have been added;
- the low-cost greedy relaxed plan extraction procedure can still be used, as the length of the non-refined plan (without the aforementioned noise) is taken to be the heuristic value.

Initial results in the DriverLog domain using this planner configuration, presented in figure 1, suggest that the use of refined plans in this manner increases the effectiveness of lookahead, providing a reduction in planning time. It can be seen that a small overhead is incurred through the analysis of the generic types in the domain, but in larger problems the reduction in planning time far outweighs this overhead. In particular, problems 16 and 19 are solved in less time, and problem 18 is solved where previously it was not (within the 30 minute time-limit to which the tests were subjected).

Selectively Introducing Delete Lists based on Generic Type Information

Another approach to making the relaxed problem more realistic would be to introduce some of the delete effects which are known to have controllable interactions within the problem, forming a ‘partially relaxed’ planning problem. In particular, if the delete effect when mobiles moved was maintained, the actions in the relaxed plan could not make use of a mobile being in two locations at once.

Two approaches are being investigated to use to solve the partially relaxed problem and return a heuristic measure:

- Using GraphPlan, as with the conventional relaxed planning problem, but handling the mutexes introduced by the added delete effects
- Using a simple partial-order approach, dealing with the mutexes by adding the necessary actions during plan time - for example, a mutex between two actions requiring a mobile to be at two locations can be dealt with by adding actions between the two to move the mobile from one location to the other.

Conclusions

This paper presented an overview of work investigating improved search guidance; with a particular focus on the identification and use of generic type information to provide better heuristic knowledge. To date, the relaxed plan extraction and lookahead techniques have been implemented and an evaluation is being performed. The implementation of the selective introduction of delete effects into the relaxed problem still in progress.

References

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Inteligence (IJCAI-95)*.

Bonet, B., and Geffner, H. 2000. HSP: Heuristic search planner. *Artificial Intelligence Magazine* 21.

Do, M. B., and Kambhampati, S. 2003. SAPA: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20:155–194.

Fox, M., and Long, D. 2001. Hybrid STAN: Identifying and managing combinatorial optimisation sub-problems in planning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 445–452. Morgan Kaufmann.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 161–170.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Long, D., and Fox, M. 2000. Automatic synthesis and use of generic types in planning. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2000)*, 196–205.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-2004)*, 150–159.