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

Andrew Coles and Amanda Smith
Department of Computer and Information Sciences,
University of Strathclyde,
26 Richmond Street,
Glasgow, G1 1XH
email: `firstname.lastname@cis.strath.ac.uk`

Abstract

This paper presents techniques for improving the quality of the search heuristics used to guide forward-chaining planning. The improvements in heuristic quality are made by using a static analysis of the planning problem to identify commonly occurring 'generic types', and providing additional heuristic guidance based on their known properties. The results presented show that the use of this additional information as part of the popular relaxed-plan heuristic improves the performance of the planner in domains with recognised 'mobile' and 'resource' generic types.

Introduction

Forward-chaining planning guided by a heuristic has proved to be an effective planning strategy in a range of 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 to the newly created problem 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.

Background

Generic Types

First proposed by Fox and Long (Long & Fox 2000) as part of the domain-analysis tool 'TIM', generic type inference is a process whereby commonly occurring entity types within a given planning problem can be identified.

TIM, a 'type-inference machine', was first conceived as a tool for automating the process of partitioning the entities present in a planning problem by analysing the functional relationships between them. To determine the types present, Finite State Machines (FSMs) are built. These are based on the transitions between the properties each entity can hold in a reachability analysis, performed forwards from the initial state in the planning problem. States are labelled according to the predicates in which the entity is participating and the parameter of the predicate to which the entity is assigned. For instance, consider the proposition:

$$(at\ x\ y)$$

From this, the FSMs for entities x and y will contain a state labelled at_1 and at_2 respectively.

A reachability analysis forwards from the initial state adds additional states to the FSMs, indicating the transitions be-

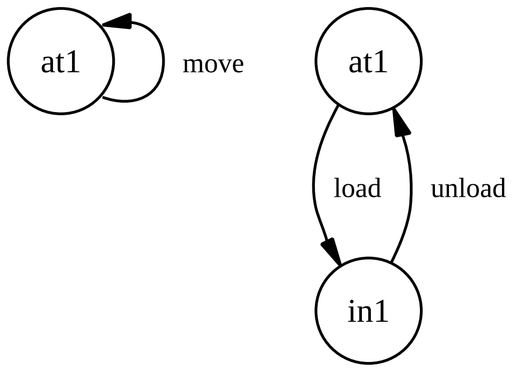

Figure 1: Finite State Machines Built in a Simple Logistics Domain (truck on the left; package on the right)

tween the properties an entity can hold arising through the application of actions. Figure 1 shows the the finite state machines built in a simple logistics problem containing one package and one truck. The left-most finite state machine corresponds to the truck, which can move from being at one location to being at another; the right-most finite state machine corresponds to the package, which can either be in the truck or at a location.

After the FSMs have been built, two entities are said to be of the same type if their associated finite state machines are identical. If additional identical trucks were added to the aforementioned logistics problem, they would each give rise to the same FSM and would thus be considered to be of the same type.

Once the basic type information was discovered by generating Finite State Machines, as described, TIM was extended to report not only the types discovered in the planning problem, but also whether each of the types corresponds to a known 'generic type' (Long & Fox 2000). A generic type is a recognisable entity type occurring in a range of planning problems; it is defined by a certain Finite State Machine topology. If the FSMs of a group of objects discovered in a given domain match that of a known generic type, TIM reports the match and provides additional information, depending on the generic type in question.

In this work, the following two recognised generic types are used:

- mobiles—entities capable of moving directly from being at one location to being at another;
- resources—a special-case of mobiles, whose map consists of a series of linearly interconnected nodes, with the location of a resource denoting its current level.

The Relaxed Planning Graph Heuristic

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 heuristic is to ignore the delete lists (negative effects) of the domain actions—a search algorithm based on that in GraphPlan (Blum & Furst 1995) is then used to solve this relaxed problem. Only a subset of the GraphPlan algorithm needs to be implemented as, due to the removal of delete effects, the planning graph does not contain mutexes.

When ignoring delete effects, 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 actions 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 problem: 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 for use 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.

The presence of a non-zero resource level has a profound effect on the profile of the search landscape when searching with the relaxed-plan heuristic. 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 weakness causing the 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 effect of 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 relaxed plans being formed which differ substantially from solution plans to the original problem. 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 locations A and E are connected through three other locations B, C and D, in a chain A-B-C-D-E. 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 for Generic Types

In many cases, one can identify actions that are logically missing from relaxed plans; that is, those 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 computing the relaxed plan heuristic, a solution to the relaxed planning problem is obtained by performing a greedy regressive search backwards from the goal within a planning graph. Each fact layer has a set of goals for which achievers must be found in an earlier action layer. Initially, the goal facts of the planning problem are added to the set of goals to be achieved in the last fact layer in the relaxed planning graph. A relaxed plan is extracted by regressing through the layers, finding achievers for each of the goals attributed to that layer. For each goal, the first recorded action achieving that goal (found when the plan graph was built) is greedily chosen as the achiever to use for the fact, and is added to the solution relaxed plan. The preconditions of the achieving action are added to the goal sets at the fact layers in which they first appeared. Plan extraction terminates when actions in the first action layer are chosen as achievers: the preconditions of these are satisfied in the initial state, therefore no additional actions are needed to achieve them.

The relaxed plan is rarely executable under the semantics of the original planning problem, where delete effects are considered. Ideally, the relaxed plan should be as close as possible to a candidate solution plan under the original semantics in order to give the best possible heuristic measure. Attempting to repair all of the cases in the plan where a deleted fact has been used, is as computationally expensive as planning itself: doing so would lead to a solution plan to the original planning problem, and thus requires an algorithm of the same complexity. Selective refinement of the relaxed plan is, however, possible by monitoring the states of generic typed objects through a simulated execution run. For both mobile and resource types, the analysis provided by TIM indicates which actions can be used to satisfy the location and resource-level properties of objects of the corresponding type.

Refining the Behaviour of Mobiles

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 a pre-planning step using the Floyd Walshall algorithm. As an example, consider the following relaxed plan built in the logistics domain:

```
0: drive-truck t1 l0-1 l0-2 c1
0: drive-truck t1 l0-1 l0-3 c1
1: load-truck p1 t1 l0-2
1: load-truck p1 t1 l0-3
2: drive-truck t1 10-2 l0-4 c1
2: drive-truck t1 10-3 l0-5 c1
3: unload-truck p1 t1 l0-4
3: unload-truck p1 t1 l0-5
```

After refinement, the relaxed plan will be as follows:

```
 0: drive-truck t1 l0-1 l0-2 c1
 1: drive-truck t1 l0-2 l0-1 c1
 2: drive-truck t1 l0-1 l0-3 c1
 3: drive-truck t1 l0-3 l0-2 c1
 4: load-truck p1 t1 l0-2
 5: drive-truck t1 l0-2 l0-3 c1
 6: load-truck p1 t1 l0-3
 7: drive-truck t1 l0-3 l0-2 c1
 8: drive-truck t1 10-2 l0-4 c1
 9: drive-truck t1 10-4 l0-3 c1
10: drive-truck t1 10-3 l0-5 c1
11: drive-truck t1 10-5 l0-4 c1
12: unload-truck p1 t1 l0-4
13: drive-truck t1 10-4 l0-5 c1
14: unload-truck p1 t1 l0-5
```

In cases where the move actions can be cleanly abstracted, such as in the logistics domain, it is then possible to post-filter the refined relaxed plan to eliminate some of the redundant drive actions. This is not possible in the general case, and although the relaxed plan is somewhat unwieldy it is still ‘less relaxed’.

Refining the Behaviour of Resources

Resources require a different refinement to mobiles. Consider the following relaxed plan segment:

```
0: sendtofree spade2 heart3 n4 n3
0: sendtofree diamond4 clubA n4 n3
```

Here, two cards are moved to the free cells. Each requires a free cell in which to store the card, and decreases the number of free cells available by one. The last two parameters of each action denote the old and new resource level for the number of free cells. However, under the ignore-delete-lists relaxation, once the first action has been applied ignoring its delete lists, the predicate stating that there are four free cells has persisted; thus, the parameters of the second action are still concerned with reducing the resource level from four to three. If the same reasoning used for mobiles was considered at this point, an action sequence to increase the resource level from three to four prior to the second action would be sought. Intuition, however, suggests a different approach here: the important aspect is the change in resource level signified by the action parameters, not the absolute values they indicate. The two actions illustrated, both of which have a -1 effect on the resource level, can be sequenced if the resource level is at least $+2$ initially. Only if it were lower than $+2$ would actions to increase the resource level be necessary.

The level of a resource as identified by TIM is denoted by an assignment to a series of ranked objects. Actions that increase the level of the resource change the variable assignment recording the resource level to a higher-ranked object; actions that decrease the level of the resource change the assignment to a lower-ranked object. All resources have limits: a maximum and minimum ranked assignment that they can take. No resource-decreasing actions are applicable in a state in which the resource level is at its minimum. If such actions are to be applied, a resource-increasing action must be applied in order to increase the resource level.

It is possible to identify where resource-level-altering actions are missing from the relaxed plan by starting with the resource level in the initiate state, and monitoring the cumulative effect of the actions within the relaxed plan on this resource level. 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 what would be off the top of the rank or off the bottom of the rank, a decreasing or increasing action needs to be inserted as appropriate. As when reasoning about mobiles, these additional actions can be inserted into the relaxed plan to make it a closer approximation to a non-relaxed plan.

In some cases, the additional action necessary is clear: for instance, it may be a simple ‘refuel’ action; in other cases, however, it is not so clear. For instance, in FreeCell if the free cells are all taken and it is not possible to move a card currently in a free cell to elsewhere, then there is no single applicable action available to increase the resource level. In other domains, there are no resource-increasing actions available to replenish a resource when it is consumed, and no remedial actions can be specified. It can, however, be noted that the relaxed plan would use non-existent resources upon execution; and a penalty of 1 added to the heuristic value taken from the relaxed plan length for each action making use of non-existant resource, thus dissuading search from pursuing such paths. 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.

Using the Refined Relaxed Plan

Having now described how a relaxed plan can be refined to make it somewhat less relaxed, the remaining issue is how such a plan should be used. In FF, the relaxed plan is used for search guidance in two ways:

- its length is taken as a heuristic goal distance estimate for each state;
- the actions chosen from the first action layer in the relaxed planning graph are used to determine the ‘helpful actions’ in each state.

Refining the relaxed plan as discussed will, however, be invariably more expensive than computing only 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; this suggests that it would be beneficial to use the plan as a source of further guidance.

YAHSP (Vidal 2004), a planner which competed at the 2004 international planning competition, extracts further information from the relaxed plan by using 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 to the current state. 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.

Performing lookahead on the refined relaxed plan, 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 only considering adding single actions, whilst allowing action sequences to be inserted where these can be determined using the generic types analysis. These action sequences potentially allow lookahead to be able to apply more of the actions within the relaxed plan, and reach a state closer to the goal; increasing its usefulness.

Results

To assess the effects on performance of making use of refined relaxed plans, the planner Marvin (Coles & Smith 2006) has been extended to make use of TIM, perform lookahead over relaxed plans, and perform the necessary heuristic refinements. To isolate the effects the use of refined relaxed plans and/or lookahead have on its performance, Marvin was configured to search in the same manner as FF: macro-actions, concurrency and symmetry-breaking were disabled.

Four domains in which TIM recognises generic types were chosen for evaluation, taken from the first, third and fifth planning competitions (McDermott 2000; Long & Fox 2003; Gerevini *et al.* 2006): logistics, DriverLog, TPP and FreeCell. The evaluation problems in each of these domains were:

- the competition problems for the DriverLog and TPP domains;
- 25 problems of increasing size in the logistics domain, the smallest with 1 aeroplane, 5 cities, 5 locations in each

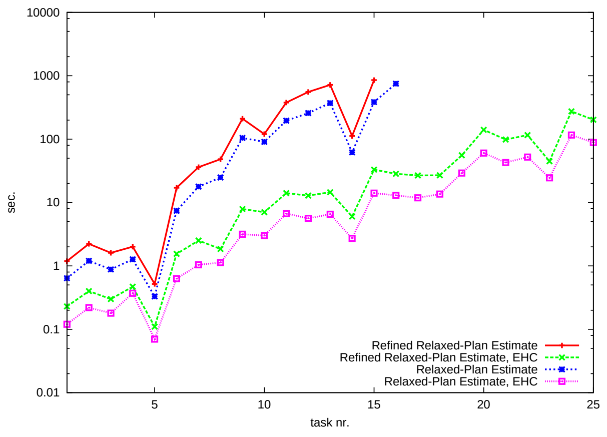

Figure 2: Time Taken to solve Twenty Five logistics Problems using Refined/Non-Refined Relaxed Plan Estimates

city and 5 packages, and the largest with 3 aeroplanes, 20 cities, 15 locations within each city and 20 packages;

- 20 full-sized problems in the FreeCell domain, generated using the IPC3 problem generator and random seeds in the range 1–20.

The performance of the planner in domains containing no recognisable generic types is unaffected by the modifications made; other than the overhead occurred due to the static analysis performed by TIM, which generally takes less than 1 second. All tests were performed on a Linux computer, with a 2.6GHz Pentium IV CPU, and were subject to a limit of 30 minutes of CPU time and 800Mb of memory.

Use of the Refined Relaxed Plan Length as a Heuristic Value

One possible use for the refined relaxed plans is to take their length as a goal-distance estimate, in place of taking the length of a non-refined relaxed plan. To assess the effect of relaxed-plan refinement on the performance of the planner when used in this manner, four configurations of the planner were used: two in which the refined relaxed-plan was used, and two in which it was not; with one configuration making use of best-first search, and another using EHC (with best-first search should this fail). The results of the planner in these configurations is presented in Figures 2 to 5.

In the logistics domain, results for which are shown in Figure 2, it can be seen that the use of a refined heuristic value increases the runtime of the planner across the board. Inspecting the number of nodes evaluated by each configuration in finding a solution, little difference is made to the proportion of the search space explored by the use of the refined heuristic; as such, given its increased computational cost, no increase in performance is realised.

In the DriverLog domain, results for which are shown in Figure 3, it can be seen that the number of problems solved by the planner is decreased by the use of the refined heuristic. On some of the larger problems there is a slight reduction in the number of nodes evaluated, but not enough to

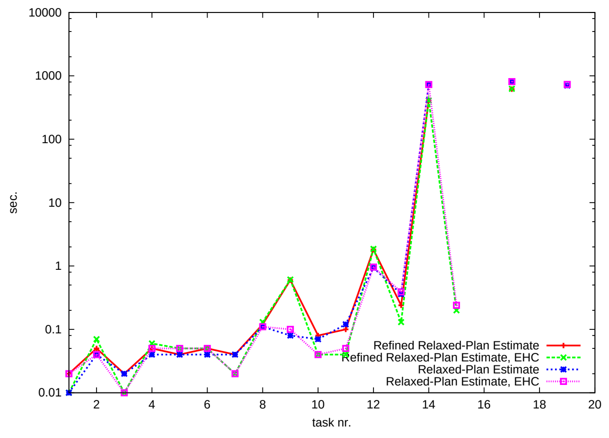

Figure 3: Time Taken to solve the Benchmark DriverLog Problems using Refined/Non-Refined Relaxed Plan Estimates

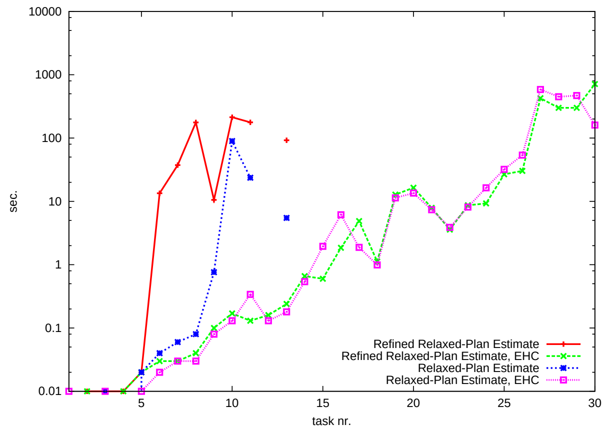

Figure 4: Time Taken to solve the Benchmark TPP Problems using Refined/Non-Refined Relaxed Plan Estimates

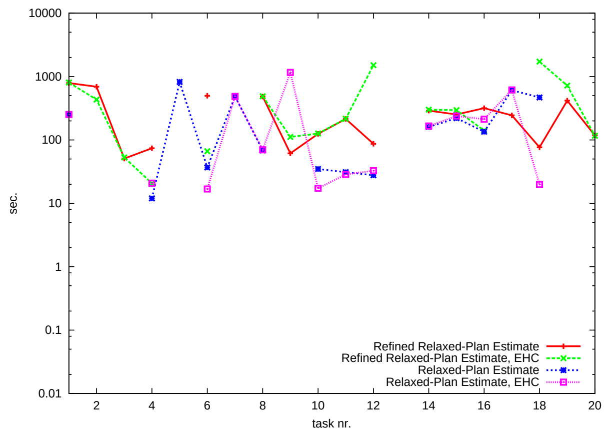

Figure 5: Time Taken to solve Twenty Full-Sized Freecell Problems using Refined/Non-Refined Relaxed Plan Estimates

make a significant impact on performance given the overheads incurred through heuristic refinement.

In the TPP domain, results for which are shown in Figure 4, it can be seen that on the larger problems there is generally a slight performance improvement through the use of a refined heuristic estimate; particularly on problems 20–29.

In the FreeCell domain, results for which are shown in Figure 5, it can be seen that the performance of the planner is improved by the use of the refined heuristic estimate: for the two best-first search configurations, 17 problems are solved when the refined heuristic is used; compared to 14 when it is not. As well as the improved performance, comparing the time taken to the number of nodes evaluated, the node evaluation time is very similar when using both the refined and non-refined heuristic estimates.

Overall, across all the domains, it appears that the the overheads incurred in calculating the refined heuristic does not produce an improvement in performance in domains with recognised 'mobile' generic types; other than slight benefits in the TPP domain. The refined relaxed plan is not being used to provide sufficient search guidance, outweighing the cost of refinement; or the length of the refined relaxed plan is not as effective as guidance search. In the FreeCell domain, however, with a recognised 'resource' generic type, an improvement in performance is realised.

Use of the Refined Relaxed Plan with Lookahead

As discussed previously, it is also possible to use the refined relaxed plan with a 'lookahead' algorithm, in which an additional successor is added to each state based on the relaxed plan from it to the goal. To assess the effect of relaxed-plan refinement on the performance of the planner when used with lookahead, five configurations of the planner were used. Two of these use the refined relaxed plan length for a heuristic value, with and without lookahead; and three using a non-refined relaxed plan length as the heuristic value, with either no lookahead, lookahead over non-refined plans or lookahead over refined plans. The results of the planner in these configurations is presented in Figures 6 to 9.

In the logistics domain the use of lookahead with the refined relaxed plan allows the goal state to be reached after the evaluation of a single node (see figure 6). When ignoring delete effects in building the relaxed plan for each package, a number of actions are included in the relaxed plan, depending on its status:

- packages in their goal location contribute no actions;
- packages in the correct city but not the correct location contribute actions to load the package into a truck in its current location (if it is not already in a truck) and unload the package from that truck in the package's goal location;
- packages in the wrong city and at an airport location contribute actions to load the package onto a plane in its current location, unload it in an airport location in the destination city, and then if necessary load/unload onto/from a truck to move it to the correct location in the destination city;
- packages in the wrong city and at a non-airport location contribute actions to load/unload the package onto/from a truck to move it to an airport location in its current city, followed by the actions contributed for packages in the wrong city and at an airport location.

Amongst these are various move actions to satisfy the preconditions relating to truck and aeroplane locations. All preconditions other than those pertaining to the locations of mobiles are satisfied in the unrefined relaxed plan. When refined, the plan becomes executable, and used with lookahead is thus able to reach a goal state. It can be seen in Figure 6 that the use of lookahead over a refined relaxed plan provides the best performance across the evaluation problems in terms of planner execution time. It also provides better performance than when lookahead over a non-refined relaxed plan is used, indicating that the extra route-planning actions which can be added due to the known generic type information are able to enhance the effectiveness of lookahead.

Figure 7 shows results for the DriverLog domain. Here, it can be seen that the best performance is obtained by the configuration in which the refined relaxed plan is used for lookahead and the non-refined relaxed plan is used to give a heuristic goal distance estimate. The performance of this configuration is closely followed by that in which the non-refined relaxed plan is used for lookahead. As in logistics, this indicates that the extra route-planning actions added to the refined relaxed plan are able to enhance the effectiveness of lookahead. Unlike the logistics domain, the refined relaxed plan is not executable under the original domain model, so more than 1 node needs to be evaluated. In general, though, it can be seen that fewer nodes are evaluated by the configurations in which lookahead over the refined relaxed plan is used.

In the TPP domain, results for which are shown in Figure 8, it can be seen that the use of lookahead over the refined relaxed plan provides substantial performance improvements over the other configurations. Lookahead over a non-refined relaxed plan evaluates similar number of nodes and exhibits runtimes similar to the two illustrated no-lookahead configurations. This indicates that the performance gains obtained by the configurations using a refined relaxed plan with lookahead are clearly attributed to the use, during lookahead, of the additional route-planning actions included. As in the DriverLog domain, the relaxed plan is not always executable under the original domain model, but it can be seen that very few nodes need to be evaluated.

In the FreeCell domain, the analysis of resource generic types leads to the identification of single actions to add to the relaxed plan in order to to increase/decrease resource levels as required. As the lookahead algorithm itself is able insert single actions to satisfy missing preconditions, the recognition of resources does not contribute actions that the lookahead algorithm would not have inserted without the recognition of resources. As can be seen in Figure 9, this leads to identical performance being obtained from the two configurations making use of lookahead over either a refined or non-refined relaxed plans whilst using the non-refined relaxed plan length goal distance estimate. As such, in this domain, the identification of resource generic types does not not affect the performance of lookahead.

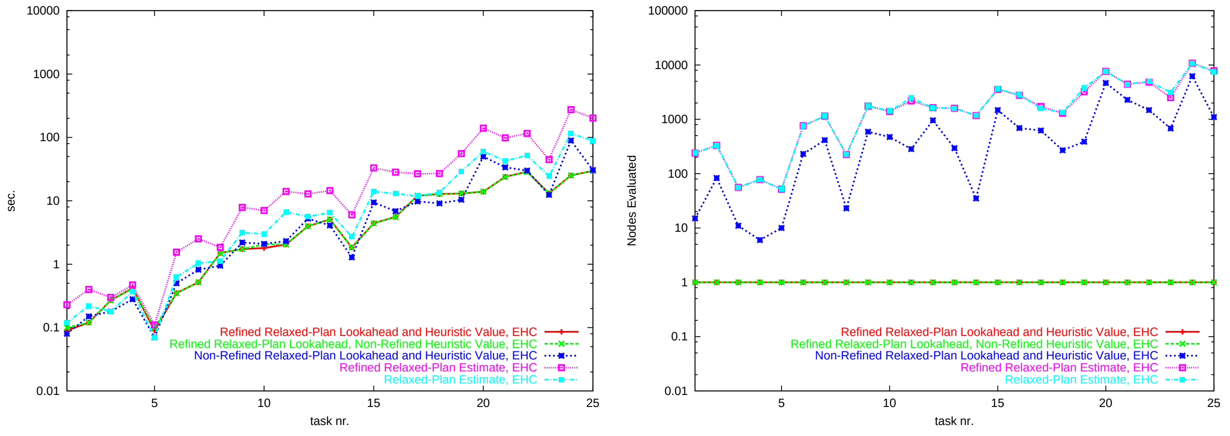

Figure 6: Time Taken and Nodes Evaluated when solving Twenty Five logistics Problems using Different Lookahead Configurations

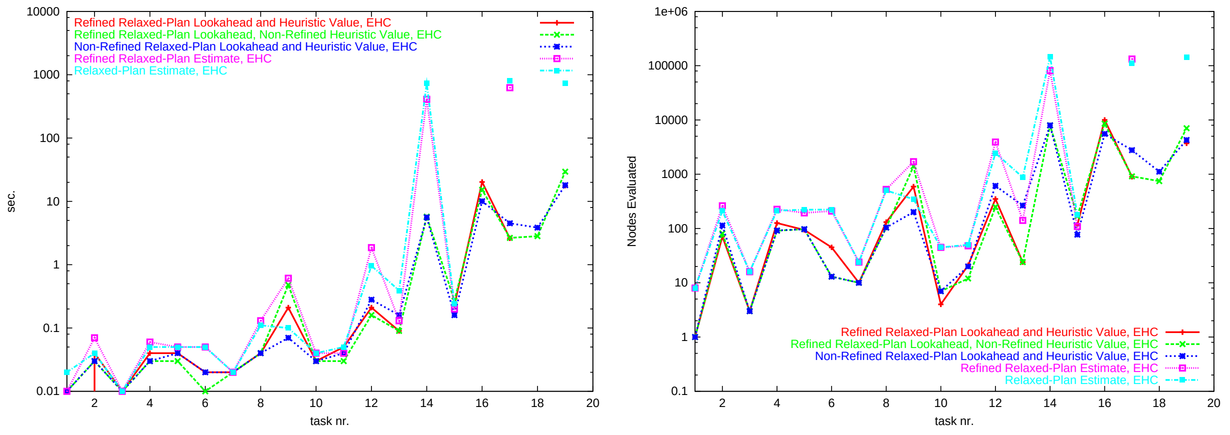

Figure 7: Time Taken and Nodes Evaluated when solving the Benchmark DriverLog Problems using Different Lookahead Configurations

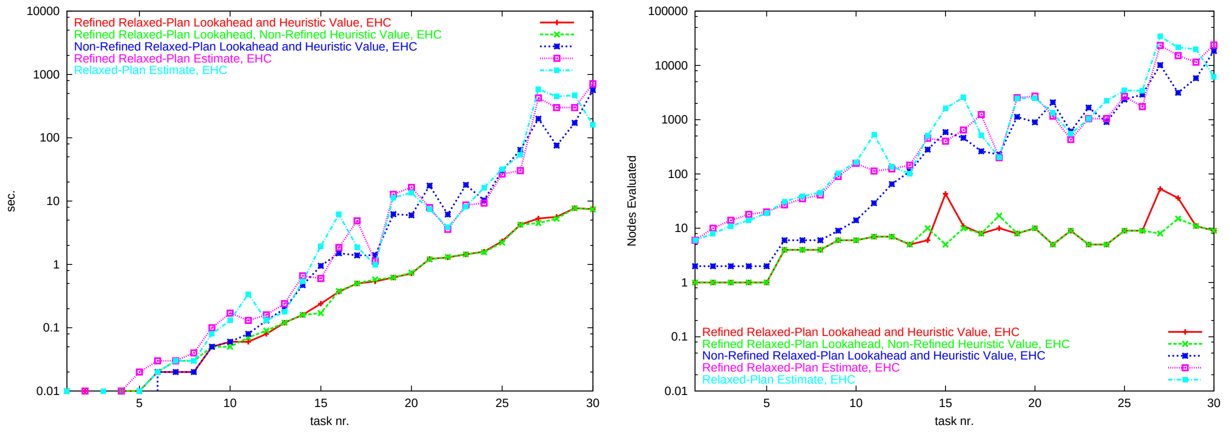

Figure 8: Time Taken and Nodes Evaluated when solving the Benchmark TPP Problems using Different Lookahead Configurations

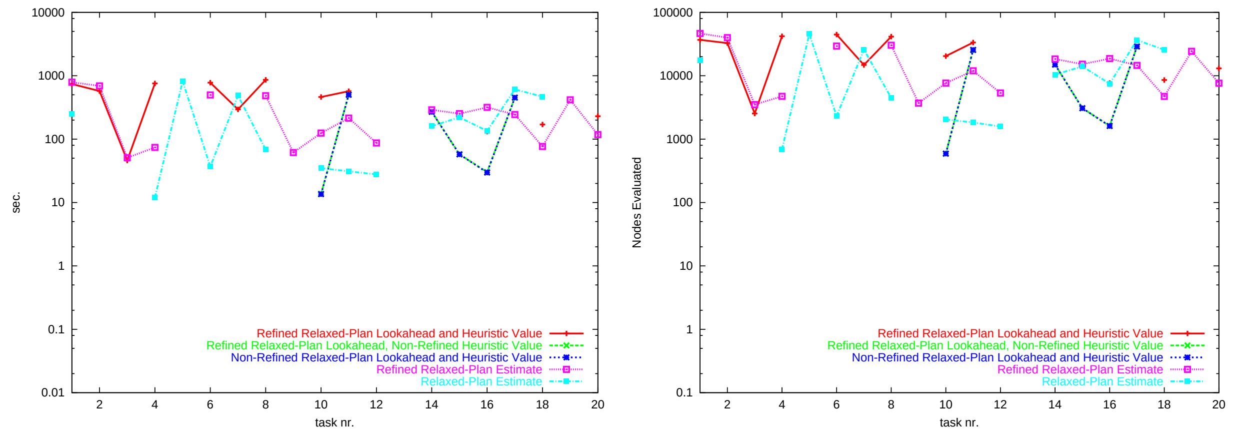

Figure 9: Time Taken and Nodes Evaluated when solving Twenty Full-Sized Freecell Problems using Different Lookahead Configurations

Overall, across all the evaluation domains, it can be seen that the identification of mobile generic types and the subsequent refinement of the of the relaxed plan allows lookahead to provide good search guidance; reducing both the number of nodes visited and the time taken to find a solution plan. As the identification of resource generic types does not suggest actions to add to the relaxed plan that would not have been included by lookahead without a refined plan, identification of resources does not affect the performance of lookahead. Nonetheless, as has been observed in the previous section, the more accurate relaxed plan length is able to provide improved search performance in these cases.

Conclusions

In this paper, it has been shown that refinements made to the relaxed plan, effected through the identification of generic types, can be used either to provide a more accurate heuristic goal distance estimate, or used as the basis of a relaxed plan over which to perform lookahead. The results obtained for these two cases indicate that:

- in general, the use of the length of the refined relaxed plan as a heuristic value when mobile generic types are identified often increases the time taken to solve each problem;
- the use of the length of the refined relaxed plan as a heuristic value in the FreeCell domain, where a resource generic type is identified, allows 17 rather than 14 of the test problems to be solved within the time limit imposed;
- the use of lookahead over the refined relaxed plan, whilst using the non-refined relaxed plan as a heuristic estimate, improves the performance of the planner—dramatically reducing the time taken to find a solution plan in some cases.

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.

Coles, A., and Smith, A. 2006. MARVIN: A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research*. Accepted for publication.

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.

Gerevini, A.; Dimopoulos, Y.; Haslum, P.; and Saetti, A. 2006. Benchmark domains and problems of IPC-5. http://zeus.ing.unibs.it/ipc-5/domains.html.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the 14th 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.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research* 20:1–59.

McDermott, D. 2000. The 1998 AI planning systems competition. *AI Magazine* 21(2):35–55.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-2004)*, 150–159.