

Extending Landmarks Analysis to reason about Resources and Repetition

Julie Porteous
J.M.Porteous@durham.ac.uk

Stephen Cresswell
S.N.Cresswell@durham.ac.uk

Dept. of Computer Science
The University of Durham
Durham, UK

September 27, 2002

Abstract

The identification of important planning subgoals, referred to as landmarks, has been shown to be useful in speeding up plan generation in a number of different planning architectures [8]. Here, we report on work in progress to extend some features of this domain analysis using landmarks. In particular we consider: the extraction of *resource abstracted landmarks*; and the identification of *landmark repetition* along with a count of the minimum number of times a landmark will need to be repeated. We illustrate our ideas with an example from a benchmark domain and consider ways in which this information could be exploited at plan time.

1 Introduction

In [8] the authors demonstrated the utility of identifying important subgoals to help speed up the planning process. These critical sub-goals, referred to as *landmarks*, are literals that must be made true in the course of any solution to a given planning problem. As an example, consider the problem from the benchmark gripper domain shown in figure 1 which we'll use as a running example throughout the paper. In this problem instance there is a single robot with 2 grippers $\{left, right\}$ and the problem is to move 4 balls $\{ball1, \dots, ball4\}$ from their initial location in *roomA* to their goal location in *roomB*. Using the analysis from [8] the set of landmarks for this problem include:

$$\{at\text{-}robby(roomA), at\text{-}robby(roomB), \dots\}$$

The analysis is also able to extract different types of orders between pairs landmarks that can then be used during plan generation to help direct search. For example, the following order is referred to as a *natural* order and it is applicable when some landmark L must be true before some other landmark L' in *all* solution plans:

$$at\text{-}robby(roomA) \leq at\text{-}robby(roomB)$$

For some domains and problems this analysis can reveal useful information but in others, such as this gripper example, it appears that some landmarks (and orders that are built on them) are being overlooked. In this gripper example for instance, we know that all of the balls must be carried at some point no matter how the goals are solved, but the decision of which gripper to use to carry them doesn't matter. It appears that in cases like this where there is an arbitrary choice of object involved in an action then landmarks are being overlooked. These objects can be seen as

a resource since the choice of which object to use is arbitrary. Part of the motivation behind the work reported here was to try and identify and abstract out these resources and in so doing both increase the number of landmarks that are identified and also the set of orders between them.

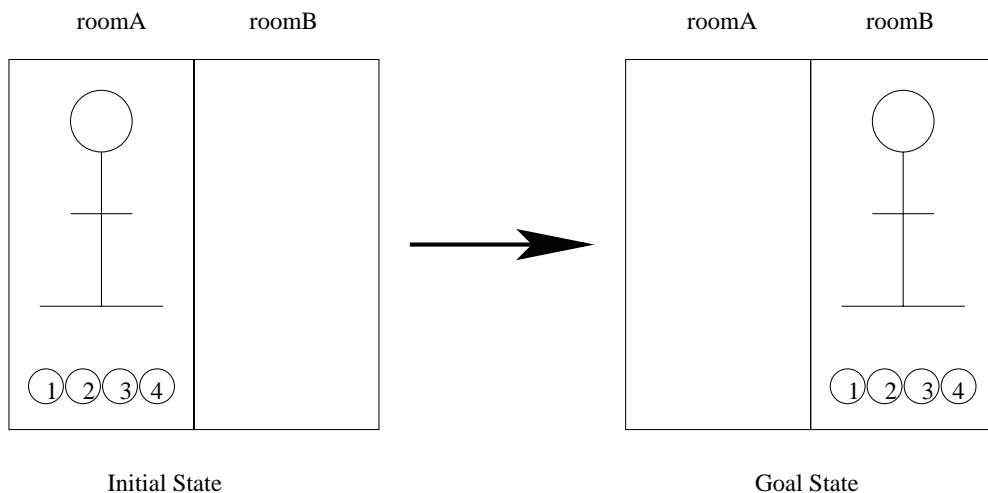


Figure 1: Gripper: Example Problem

In addition, we were also interested in looking further into the presence of *cycles* in the graphs of landmarks orders that were reported in [8]. Their approach was to simply remove any cycles that were detected but the cycles are interesting and potentially useful because they suggest certain landmarks that must be achieved more than once in the course of a plan and it may be possible to exploit this information during plan construction. We were intrigued by problems where plans frequently involve loops of repeated landmarks (or sequences of landmarks) yet which didn't result in cycles in the landmark orders. As an example, a possible solution plan to the gripper problem from figure 1 is as follows:

```
< pick(ball4, roomA, left), pick(ball3, roomA, right), move(roomA, roomB),
  drop(ball4, roomB, left), drop(ball3, roomB, right), move(roomB, roomA),
  pick(ball2, roomA, left), pick(ball1, roomA, right), move(roomA, roomB),
  drop(ball2, roomB, left), drop(ball1, roomB, right) >
```

Here we can see that any solution plans would necessarily involve repetition of a number of landmarks. For instance, a precondition of picking up any of the balls is to have *at-robby(roomA)*. This is true in the initial state and must be made true at least once more in order to move all of the 4 balls (there are of course longer plans where it is made true many more times) so here we would like to be able to identify that landmark *at-robby(roomA)* must be repeated and that there is a minimum bound of 2 on the number of times it must be true. Hence, we were also motivated to look to extend the landmarks analysis to identify landmark repetition and to get a lower bound on the number of repetitions.

The organisation of the remainder of the paper is as follows: in section 2 we discuss the extension of landmarks analysis to abstract out resources to reveal previously overlooked landmarks; and in section 3 we discuss the extension of the landmarks analysis to extract information on landmark repetition. Finally in section 4 we consider how this information can be exploited during plan construction.

2 Extending landmarks analysis to reveal resources

The method of landmark extraction presented in [8] performs a backwards search through a relaxed planning graph (RPG) (relax the planning graph by ignoring all delete lists, then build Graphplan’s planning graph, chaining forward from the initial state of the task to a level where all goals are reached [5]). During this search, landmarks are literals in the intersection of the preconditions of sets of actions that achieve a goal. For the gripper problem shown in figure 1 the set of landmarks extracted are:

$$\{at\text{-}robby(roomB), at(ball1, roomB), \dots, at(ball4, roomB), free(left), free(right), \\ at(ball1, roomA), \dots, at(ball4, roomA), at\text{-}robby(roomA)\}$$

But here we can see that landmarks analysis has overlooked some potentially useful literals because there is an arbitrary choice of an object involved in an action. For instance, the *carry* literals can’t appear as landmarks, because each ball may be carried in either gripper. If the robot had only a single gripper, then we could say with certainty that each ball must be carried with that single gripper in all solution plans and hence identify these literals as landmarks. However, when there are multiple grippers there is choice over which gripper to use to carry each ball, although we still know with certainty that each ball must be carried at some point. We argue that the choice of object at this point represents a *resource* and it can be abstracted from the landmark since the allocation of the resource is not critical at this stage. It is worth pausing at this point to consider exactly what we mean by a resource. The definition of what constitutes a resource is still a research issue but using the classification from [6] we can classify the type of resource that we are interested in as being *Qualitative*. This means that the resource is in fact represented by the states of distinct objects and it is the state of an object which is the resource. For example in the gripper problem, the gripper objects, *left* and *right*, can be seen as resources and they can occupy different states: they can be either available, *free(left)* and *free(right)*; or not available, *carry(ball1, left)*, *carry(ball1, right)*, \dots , *carry(ball4, left)* and *carry(ball4, right)*.

We would like to be able to identify these *resource abstracted landmarks* which, as the name suggests, are landmarks where the resource objects have been abstracted out. For the gripper problem this would mean the extension of the set of landmarks to include:

$$carry(ball1, X), carry(ball2, X), carry(ball3, X), carry(ball4, X)$$

where the variable X represents the gripper resource with the choice of binding of $\{left, right\}$ which can be deferred till plan construction¹.

2.1 Extracting (resource abstracted) landmarks

Figure 2 shows an outline algorithm that we have implemented to extract landmarks from a specification of a planning problem. In this section we’ll go through the main points of the algorithm, highlighting key features and illustrating them with reference to the gripper problem that we’re using throughout the paper.

The procedure maintains two sets: *Candidates* which initially contains the top level problem goals each as a singleton set; and *Landmarks*, initially set to $\{\}$ and where extracted landmarks are collected. Its important to note that in this algorithm we treat every landmark as a disjunctive set – these sets may be singleton, containing a single landmark, or else the set of literals represents a resource abstracted landmark. As these sets are updated any disjunctive sets which are supersets of others are filtered out.

¹In essence, what we are collecting here are disjunctive sets of literals which represent the alternatives from which a choice must be made. We would like to acknowledge the contribution of Laura Sebastia of the Technical University of Valencia, Spain, for an earlier implementation that extracted disjunctive sets and also for valuable discussion.

procedure: extract_landmarks $Landmarks \leftarrow \{\}$ add each of the top-level goal literals (as a singleton set) to $Candidates$ **repeat** until $Candidates$ is empty:

1. remove a landmark L from $Candidates$
 2. add L to $Landmarks$
 3. construct RPG, excluding L
 4. from RPG, extract the set of $first_achievers(L)$
 5. compute the intersection I , of preconditions of $first_achievers(L)$
 6. add each literal in I to $Candidates$
(if not already in $Landmarks$ or $Candidates$)
 7. compute same-type disjunctions from preconditions of $first_achievers(L)$ that are not in I
 8. add disjunctions to $Candidates$
(if not already in $Landmarks$ or $Candidates$)
-

Figure 2: Resource Abstracted Landmark Extraction

For the gripper problem $Candidates$ would initially contain: $\{\{at(ball1, roomB)\}, \{at(ball2, roomB)\}, \{at(ball3, roomB)\}, \{at(ball4, roomB)\}\}$ The algorithm continues until there are no more $Candidates$ to test and at that point $Landmarks$ contains the set of landmarks that have been extracted.The procedure processes each landmark in $Candidates$ in turn and continues until this set becomes empty. As each set of landmarks, L , is removed from $Candidates$ it is marked as visited and an RPG is constructed. There are a couple of points to note about this RPG:

- the RPG is constructed until a *fixed point* is reached where no new literals or actions can be added. This is different to the method outlined in [8] where the RPG is constructed until the first level that achieves the current goal of interest. By extending the RPG to the fixed point we ensure that all ways of achieving a literal are considered, rather than just the ones that are achieved earliest in the RPG and this ensures the extraction of valid landmarks without the need for further verification;
- for any L we only ever build an RPG that *excludes* L , denoted RPG_L . To exclude L we simply exclude all actions that add any of the literals in L .

We then use this RPG_L to get the $first_achievers$ of L which we define as:the set of actions that add at least one of the literals in L and whose preconditions are all reachable in RPG_L The result is that $first_achievers$ returns a subset of all the possible achievers of L , made up of only those actions whose preconditions are reachable in RPG_L .So suppose as an example that in step 1 we select L as $\{at(ball1, roomB)\}$ then the set of $first_achievers(L)$ output in step 4 would be: $\{drop(ball1, roomB, left), drop(ball1, roomB, right)\}$

with preconditions as follows:

 $drop(ball1, roomB, left) : carry(ball1, left), at-robby(roomB)$ $drop(ball1, roomB, right) : carry(ball1, right), at-robby(roomB)$

Steps 5 and 6 take the intersection, I , of the preconditions of the first achievers of L and each literal in I represents a single fully ground landmark which is added to $Candidates$ as a singleton – intuitively we can see that there is no choice of resources here since each literal is common to *all* ways of achieving L . One point to note here is that in step 6 we check whether any $i \in I$ are initial state literals. For any i that are, then they are added directly to the set $Landmarks$ and no further processing of them is required.

Continuing on with the example, we find the intersection of the *first_achievers* of L is:

$$\{at\text{-}robby(roomB)\}$$

and then the set $Candidates$ is updated to become:

$$\{\{at(ball2, roomB)\}, \{at(ball3, roomB)\}, \{at(ball4, roomB)\}, \{at\text{-}robby(roomB)\}\}$$

In steps 7 and 8 some processing may be required to compute *same-type* disjunctions from the preconditions of *first_achievers* that are not in I . Members of each *same-type* disjunction must satisfy the following conditions:

- the literals in the set must have the same name and the same number of arguments
- each disjunctive set, D , must contain a literal from the preconditions of *each* action in *first_achievers*

which ensures that we have considered all possible choices in the context of L .

At this stage in our example, the preconditions of the *first_achievers* that aren't in I are:

$$\{carry(ball1, left), carry(ball1, right)\}$$

and since these are both of the same type and contain a literal from each action in *first_achievers* no further processing is necessary and this disjunctive set is added directly to $Candidates$ which becomes:

$$\{\{at(ball2, roomB)\}, \{at(ball3, roomB)\}, \{at(ball4, roomB)\}, \{at\text{-}robby(roomB)\}, \\ \{carry(ball1, left), carry(ball1, right)\}\}$$

In this example problem there is only a single argument that represents the presence of a resource. It may happen that there are multiple resources and multiple same type disjunctions and they can be handled in the same way.

For the rest of the paper we will represent the resource abstracted arguments of landmarks as variables denoted by upper case letters. So, the final set of landmarks extracted for the gripper problem are as follows (the domain of variable X is $\{left, right\}$):

$$\{\{at(ball1, roomA)\}, \{at(ball2, roomA)\}, \{at(ball3, roomA)\}, \{at(ball4, roomA)\}, \\ \{free(X)\}, \{at\text{-}robby(roomA)\}, \{carry(ball1, X)\}, \{carry(ball2, X)\}, \{carry(ball3, X)\}, \\ \{carry(ball4, X)\}, \{at\text{-}robby(roomB)\}, \{at(ball1, roomB)\}, \\ \{at(ball2, roomB)\}, \{at(ball3, roomB)\}, \{at(ball4, roomB)\}\}$$

In this example the gripper was identified as a resource with respect to each of the $at(ball1, roomB)$, \dots , $at(ball4, roomB)$ goals but it is possible for the role of an object to change in the course of a problem. For example, if the gripper problem was extended to include $ball5$ and an additional goal $carry(ball5, left)$ then the gripper object would still be a resource in the context of the $at(ball1, roomB)$, \dots , $at(ball4, roomB)$ goals but it would not be a resource in the context of the new goal. So an important feature of the resources that are identified is that they are local with respect to a particular landmark or top-level goal. We return to discussion of the types of resources in section 4.

2.2 Finding orders between (resource abstracted) landmarks

In the previous section we discussed a method for extracting resource abstracted landmarks and here we build on this to extend the definitions of some of the orders that can be imposed between them. This involves an extension to disjunctive sets of some of the orders discussed in [8] and also introduction of a new order that we exploit in reasoning about landmark repetition.

- **Natural orders**

When *all* solution plans to a problem order a landmark L before some other landmark L' then we call this a natural order, expressed as $L \leq_n L'$. We can identify these orders between landmarks using the procedure given in figure 2 by testing which landmarks L' become unreachable in RPG_L . If L' is unreachable in RPG_L then they are ordered $L \leq_n L'$. In the gripper problem, an example is:

$$\text{at-robby}(\text{room}A) \leq_n \text{at-robby}(\text{room}B)$$

- **Necessary orders**

For our purpose these are defined as natural orders $L \leq_n L'$ for which L is necessarily true in the state immediately before L' is achieved and we denote them using \leq_{ne} . We identify orders where this next-state property holds the first time L' is achieved by checking that L is a precondition of all *first_achievers* of L' . In the gripper problem, an example is:

$$\text{carry}(\text{ball}1, X) \leq_{ne} \text{at}(\text{ball}1, \text{room}B)$$

- **Destructive necessary orders**

For a pair of resource abstracted landmarks L and L' , a destructive necessary order is a necessary order, $L \leq_{ne} L'$, for which all achievers of L' delete some disjunct in L , denoted using \leq_{dn} . We have found these orders useful for reasoning about repeated landmarks and we will discuss this further in section 3. A gripper example is:

$$\text{at-robby}(\text{room}A) \leq_{dn} \text{at-robby}(\text{room}B)$$

- **Weakly reasonable orders**

These were introduced in [8], and impose an order between pairs of literals to avoid wasted effort during plan generation. Identification of them involves reasoning about inconsistency between landmarks and so we must extend the definition from [8], to cover what it means for two disjunctive sets of literals to be inconsistent. This can be stated as:

two disjunctive set landmarks L_1 and L_2 are inconsistent if, for any $d_1 \in L_1$, and for any $d_2 \in L_2$, we have *inconsistent*(d_1, d_2)

or in other words, it is not possible to find a pair d_1 and d_2 that are consistent. An example from the gripper problem is:

$$\text{carry}(\text{ball}1, X) \leq_{wr} \text{at-robby}(\text{room}B)$$

which reflects the fact that, given the initial state for this problem where both *robby* and *ball1* are in *roomA*, we can avoid wasted effort by trying to achieve *carry(ball1, X)* before moving *robby* to *roomB*

For the gripper problem that we have been using throughout the paper the graph of orders we obtain is shown in figure 3. For clarity in the figure the disjunctive orders have been abbreviated (so for example the set $\{\text{carry}(\text{ball}1, \text{left}), \text{carry}(\text{ball}1, \text{right})\}$ is shown as *carry(ball1, X)*).

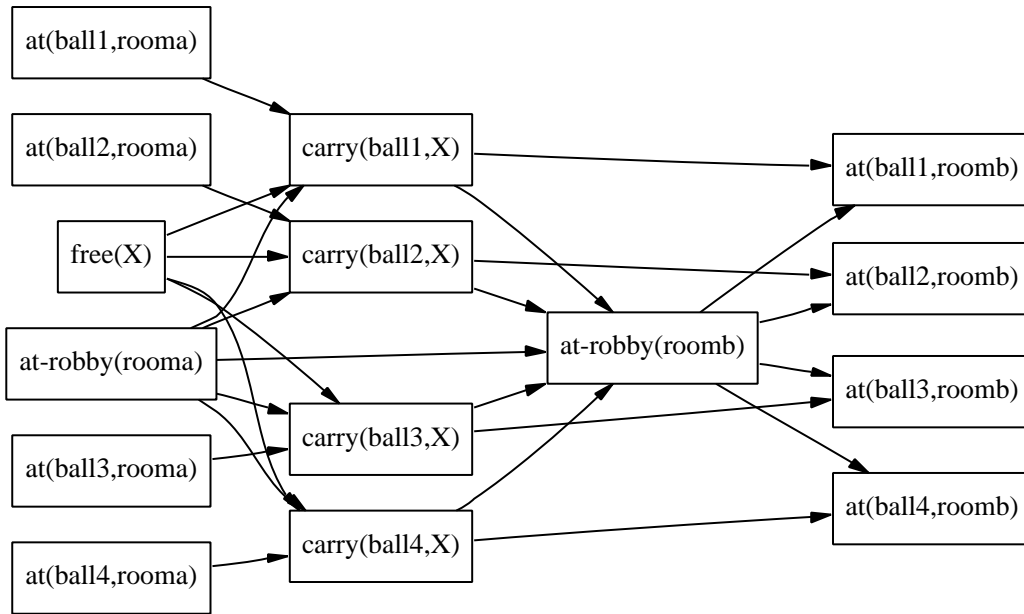


Figure 3: Gripper: Graph of landmark orders

3 Identifying and Counting Landmark Repetition

This work was in part motivated by a desire to identify landmarks that must be made true more than once during solution of a given planning problem. If this information could be computed cheaply in a phase of static domain analysis then it could be useful during plan construction. We have had some success towards this goal and in this section we report on our work in progress in this area.

Our approach is based on classifying various inconsistencies in the graph of landmark orders by matching a set of rules locally in the landmarks graph. This inconsistency detection combines information from both landmarks analysis, which is insensitive to resource limitations, and the TIM analysis of [4], which is insensitive to relations between specific objects. We start by discussing a novel use of TIM analysis and then discuss how we have exploited this information in identification of landmark repetition.

3.1 Use of TIM to count minimum coverage

Access to the TIM analysis of [4] is available via the TIM API [1] and provided within this is the *TIMinconsistent* function that takes as input a pair of literals and returns true if the set are inconsistent (i.e. can not be made true in the same state) and false otherwise. We have implemented a function, called *min-coverage*, that is an extension to this: it is input a set of landmarks and outputs a lower bound on the number of states that would be required to cover all the literals in the set, or in other words the minimum coverage. For example, if a set of literals are consistent then the lower bound is 1 but if they are not consistent, like the 4 *carry* literals that we saw in the gripper problem, then the lower bound is 2 (reflecting the resource width limit here of the number of *carry*'s that can be true in parallel since there is a single robot with 2 grippers and 4 balls to be carried).

The method for obtaining this count is to look at the TIM property spaces to get the maximum number of instances of a particular property that can be true in any state. In the gripper problem

that we saw earlier in the paper we can reason about the number of times that *carry* can appear in a single state: we know that the maximum number of balls that can be carried at any one time is 2 balls (given the capacity of the gripper). Since the set of literals that are being tested for consistency requires 4 distinct instances of *carry*, then we know that at least 2 distinct states will be required.

3.2 Propagation Rules for landmark repetition

The general idea is to match one of a number of propagation rules against the graph of (resource abstracted) landmark orders. Any matches that are found yield information about sets of landmarks to test, to obtain a count of landmark repetition. So far we have identified two propagation rules which we describe in turn.

Propagation Rule 1

We match to the landmarks graph as shown in figure 4, where L is some (resource abstracted) landmark and where P_1, \dots, P_n are *all* landmarks that are ordered before L such that $P_1 \dots P_n$ must all be true in the state immediately before L . *Necessary* orders always satisfy this condition. *Weakly reasonable* orders satisfy the condition with some additional restrictions not described here.

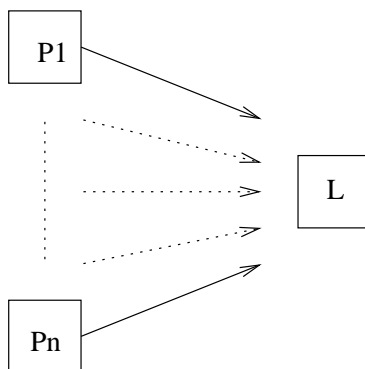


Figure 4: Propagation Rule 1

Having identified the set $P = \{P_1, \dots, P_n\}$ from the landmarks graph we use *min-coverage* to get a count of the minimum number of distinct consistent states covering P and we assert that the number of times L must be true in any solution plan to this problem, N_L , is:

$$N_L \geq \text{min-coverage}(\{P_1, \dots, P_n\})$$

As an example of the use of this rule consider the graph of orders between landmarks in the gripper problem shown in figure 3 and where L and P are as follows:

$$L = \text{at-robby}(\text{room}B)$$

$$P = \{\text{carry}(\text{ball}1, X), \text{carry}(\text{ball}2, X), \text{carry}(\text{ball}3, X), \text{carry}(\text{ball}4, X), \text{at-robby}(\text{room}A)\}$$

The output of *min-coverage*(P) is 2 so we can reason that *at-robby*(*room* B) must be true at least 2 times in the course of all solution plans to this problem.

In contrast consider the application of this when L and P are:

$$L = \text{at}(\text{ball}1, \text{room}B)$$

$$P = \{\text{carry}(\text{ball}1, X), \text{at-robby}(\text{room}B)\}$$

Here the result of $\text{min-coverage}(P)$ is 1 (reflecting the fact that they can all be made true simultaneously – recall that $\text{carry}(\text{ball1}, X)$ is resource abstracted so the 2 possible sets of literals that are considered $\{\text{carry}(\text{ball1}, \text{left}), \text{at-robby}(\text{roomB})\}$ and $\{\text{carry}(\text{ball1}, \text{right}), \text{at-robby}(\text{roomB})\}$ both of which are consistent). The output of the reasoning is that there is no requirement to repeat the landmark $\text{at}(\text{ball1}, \text{roomB})$.

Propagation Rule 2

We match to the landmarks graph as shown in figure 5, where L is some (resource abstracted) landmark and where the set $S = \{S_1, \dots, S_n\}$ contains *all* landmarks that are ordered directly after L by *destructive necessary* orders, or in other words, where L appears as both a precondition and a delete effect of every possible achiever of each landmark in S . The reasoning for this rule is as follows: any achieving action for a landmark in S necessarily consumes an instance of L . Therefore the minimum number of times L must occur is determined by the minimum number of achieving actions required to cover all of the landmarks in S . We call this quantity $\text{min-achievers}(S)$.

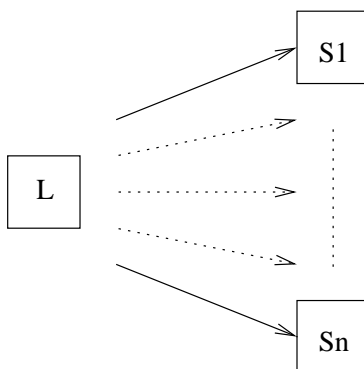


Figure 5: Propagation Rule 2

Having identified the set $S = \{S_1, \dots, S_n\}$ from the landmarks graph, $\text{min-achievers}(S)$ is then computed and this yields a count of the minimum number of distinct achievers required to obtain landmarks in S , and we assert that the number of times L must be true in any solution plan to this problem, N_L , is:

$$N_L \geq \text{min-achievers}(S)$$

As an example of the use of this rule consider the graph of orders between landmarks in the gripper problem shown in figure 3 and where L and S are as follows (the domain of X is $\{\text{left}, \text{right}\}$):

$$L = \{\text{free}(X)\}$$

$$S = \{\text{carry}(\text{ball1}, X), \text{carry}(\text{ball2}, X), \text{carry}(\text{ball3}, X), \text{carry}(\text{ball4}, X)\}$$

The output of $\text{min-achievers}(S)$ is 4, so we can reason that $\text{free}(X)$ must be true a minimum of 4 times in any solution plan.

4 Conclusions and Future Work

In the work to date we have extended work on landmarks from [8] so that we are now able to extract information about landmarks that must be repeated in any solution plan for an input problem. We have an implementation that is able to give a minimum bound on the number of times that a given landmark will need to be made true during any solution plan. This method for counting landmark repetition relies on being able to abstract out local resource use and we have been able to extend our extraction and ordering of landmarks to cover such resource abstraction.

One way in which we intend to progress this work is to look at automatic identification and abstraction of resources in planning problems. For example, in the `RealPlan` system of [9] the general idea is to de-couple resource allocation from planning and to handle resources in a separate scheduling phase. They have shown that this approach can work well in problems where the performance of some planners degrades in the presence of certain types of resources (they consider `Graphplan` [2], `FF` [5] and `UCPOP` [7]). In `RealPlan` global resources are identified by the domain modeller which places an extra burden on them and also ignores local resources which are important since an object may play different roles during the lifetime of a plan. As an example, consider a truck in a logistics domain which can be a resource in the context of delivering a particular package but isn't a resource in the context of a final goal of getting it to a depot once the package has been delivered (for further discussion see [6]). We intend to extend our reasoning to help identify such resources. This is likely to involve reasoning about actions that delineate parts of the plan that are resource locked (such as `pick` and `drop` in the gripper domain, between which the choice of the gripper resource is locked) and hence extending the reasoning about repetitions from distinct landmarks to sets of landmarks or actions.

Another way to take this work forward is to use the information about repeated landmarks (and hence repeated actions) to derive a lower bound on the number of actions that must occur in a plan. There may be some interesting issues here, as what we will extract directly for each landmark is a minimum number of instances drawn from a set of achieving actions. This could then be used in different planning architectures: for example in heuristic forward search planners, such as `FF` [5] and `HSP` [3], this may lead to improvements to the heuristic estimate of a states' utility. This may be particularly true in the presence of resources when the performance of such systems degrades (for discussion of this see [9]). In future we intend to look at the use of this landmark count information in a range of different planning architectures.

References

- [1] The TIM API is available on line from:
<http://www.dur.ac.uk/computer.science/research/stanstuff/planpage.html>.
- [2] A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90, 1997.
- [3] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 2001.
- [4] Maria Fox and Derek Long. The Automatic Inference of State Invariants in TIM. *Journal of Artificial Intelligence Research*, 9, 1998.
- [5] J. Hoffmann and Nebel B. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14, 2001.
- [6] D. Long, M. Fox, L. Sebastia, and A. Coddington. An examination of resources in planning. In *Proceedings of the 19th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2000)*, 2000.
- [7] J. Penberthy and D. Weld. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *Proceedings of AAAI '94*, 1994.
- [8] J. Porteous, L. Sebastia, and J. Hoffmann. On the Extraction, Ordering and Usage of Landmarks in Planning. In *Proceedings of the 6th European Conference on Planning (ECP '01)*, 2001.
- [9] B. Srivastava, S. Kambhampati, and M. Do. Planning the Project Management Way: Efficient Planning by Effective Integration of Causal and Resource Reasoning in `RealPlan`. *Artificial Intelligence*, 2001.