A SAT Compilation of the Landmark Graph

Vidal Alcázar
Universidad Carlos III de Madrid
Avenida de la Universidad, 30
28911 Leganés, Spain
valcazar@inf.uc3m.es

Manuela Veloso
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
veloso@cmu.edu

Abstract

Landmarks are subgoals formed by sets of propositions or actions that must be achieved at some point in a planning task. These landmarks have a series of ordering relations between them that form what is known as the landmark graph. Previous works have used information from this graph with several purposes; however, few have addressed some of the shortcomings of the current representation of the graph, like landmarks having to be true at several time steps. In this work we propose a SAT encoding of the landmarks graph whose solution represents a more informative version of the original graph.

Introduction

Landmarks are disjunctive sets of propositions of which at least one component must be achieved or executed at least once in every solution plan to a problem (Hoffmann, Porteous, and Sebastia 2004). Currently landmarks are a very prominent line of research in automated planning, as the success of landmark-based planners like LAMA (Richter and Westphal 2010) shows. Most approaches have focused on using landmarks to partition the problem (Hoffmann, Porteous, and Sebastia 2004; Sebastia, Onaindia, and Marzal 2006) and to derive heuristics used in forward search planners (Richter and Westphal 2010; Karpas and Domshlak 2009; Bonet and Helmert 2010).

Finding the complete set of landmarks or even just proving that a proposition is actually a landmark is PSPACE-complete (Hoffmann, Porteous, and Sebastia 2004). However, current methods can efficiently compute a subset of the landmarks of the task based on a delete-relaxation representation of the problem. A series of partial orders between propositions can be computed with similar techniques too, which applied to landmarks are used to build the landmark graph. The landmark graph is the base of many of the techniques that employ landmarks, as the order in which landmarks should be achieved is often as important as finding the landmarks themselves.

Even though the landmark graph provides important information it still has several shortcomings. First, the partial orderings may not be enough to come up with a reasonable total order, which is critical for things such as partitioning the problem into smaller subproblems. Second, landmarks appear only once, even when it is clear that they must be achieved several times, like when they are causal preconditions of other landmarks that cannot be true at the same time. For example, in the well-known *Blocksworld* domain, *(arm-empty)* appears only once despite being necessary before every possible *(holding x - block)*, which cannot be true at the same time as any other proposition derived from the predicate *holding*. Third, the exploitation of the cycles and the unsound orderings in the graph is still unclear. In fact, most techniques that use landmarks are designed to be applied to acyclic graphs, so cycles are usually removed discarding unsound edges first before any kind of search begins.

An important remark is that the concept of time in a total order setting is absent in the landmark graph. In order to introduce time, landmarks must be able to appear as needed in different time steps. For this, they must be labeled with some sort of time stamp. The planning graph (Blum and Furst 1997) that is often used to represent the planning task does specifically this, as every proposition and action is represented several times by nodes labeled with the level they appear in. Hence, landmarks can be represented as several nodes, one per different possible time step. There is a similar relationship between the orderings in the landmark graph and the different edges of the planning graph: orderings are constraints that represent causal relationships of precedence between landmarks, and edges in the planning graph are constraints of either action-proposition causality or mutual exclusivity.

One of the most popular ways of exploiting the information contained in the planning graph is encoding it into a SAT problem (Kautz and Selman 1999). Using a SAT solver to find a solution to the encoding allows to find a corresponding solution plan to the problem or to prove that there is none for a given number of parallel time steps. The encoding consists in creating a variable per node in the planning graph and a clause per constraint between nodes. Given the similarity between the planning graph and the time-stamped landmark graph, this can also be done for the latter in order to obtain an enhanced version of the landmark graph that allows the possibility of landmarks being required at different time steps and that represents a more consistent total order than the one represented by the original graph. In addition, binary mutual exclusive relationships (mutex) will be introduced explicitly as another way of enforcing temporal constraints.

In this work we propose a SAT compilation for the land-

mark graph. First, some background regarding automated planning and landmarks will be given. Afterwards previous related work will be analyzed making emphasis on how the information of landmark graph has been exploited. The process of encoding the landmark graph into a SAT problem will be described next, going into detail for every feature relevant to the compilation. Some experimentation will be done to demonstrate the viability of the approach and finally some conclusions will be drawn and a few remarks on future lines of research will be presented.

Background

Automated planning can be described as the task of finding an ordered sequence of actions (commonly referred to as a plan) that achieves a set of goals from a given initial state. In this work only propositional planning is considered. A standard formalization of a planning problem is a tuple *P=(S,A,I,G)* where *S* is a set of propositions, *A* is the set of grounded actions derived from the operators of the domain, $I \subseteq S$ is the initial state and $G \subseteq S$ the set of goal propositions. The actions that are applicable depend on several propositions being true and their effects can make propositions true of false, which is commonly known as *adding* and *deleting* the propositions, respectively. This way an action would be defined as a triple {*pre(a), add(a), del(a)*} in which $a \in A$ and *pre(a), add(a), del(a)* $\in S$. Actions can have an associated cost; in this work only non-negative cost actions are used.

A planning graph is a directed leveled graph used to represent the constraints of a propositional planning problem. There are two kind of levels, *proposition levels* and *action levels*. Both kind of levels are alternated. Every node in the graph is labeled with the proposition or action it represents and the level (which actually represents a time step) it appears in. The first level is composed by the propositions $s \in I$. Subsequent levels contain propositions and actions that could be true at that level. For instance, the actions in level 2 are those applicable from the initial state, and the propositions in level 3 are the propositions that appeared in level 1 plus those achieved by the actions in level 2. Edges in the graph are derived from the triple {*pre(a), add(a), del(a)*} that defines actions. There are also binary mutual exclusivity (mutex) relationships between nodes in the same level, which represent that both nodes cannot be true at the same time. Edges and mutexes represent the constraints of the planning problem.

Landmarks were initially defined as propositions that had to be true at some point in every solution plan to a problem. This concept was extended later to disjunctive sets of propositions and actions that had be respectively achieved or executed at some point (Richter and Westphal 2010) and more recently also to conjunctive sets of propositions (Keyder, Richter, and Helmert 2010). A general definition of landmark follows:

Definition 1 *A landmark is a logical formula* L *over either* S *(proposition landmark) or* A *(action landmark). Every solution plan must satisfy every action landmark. For each proposition landmark, at least one state in every sequence of states generated by a solution plan must satisfy it.*

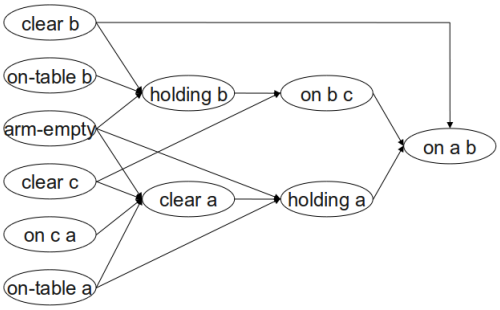

Figure 1: Simplified landmark graph of the Sussman's anomaly.

Orderings between facts are relations between two sets of propositions which represent the order in which they should be achieved in a given problem. There are the following orderings:

- Natural ordering: A proposition *a is naturally ordered before b* if *a* must be true at some time before *b* is achieved
- Necessary ordering: A proposition *a is necessarily ordered before b* if *a* must be true one step before *b* is achieved
- Greedy-necessary ordering: A proposition *a is greedy-necessarily ordered before b* if *a* must be true one step before *b* when *b* is first achieved
- Reasonable ordering: A proposition *a is reasonably ordered before b* if, whenever *b* is achieved before *a*, any plan must delete *b* on the way to *a*, and re-achieve *b* after or at the same time as *a*

Previous works classify reasonable orders as unsound (Hoffmann, Porteous, and Sebastia 2004), as not every solution plan has to achieve *a* before *b* if *b* is ever achieved. Obedient-reasonable orderings are a special case of reasonable orderings which arise when all previously computed reasonable orders are assumed to hold, although they will not be considered to this work.

The landmark graph is the directed graph composed by the proposition landmarks of a problem and the orderings between them. It is not acyclic, as necessary and reasonable orderings can induce cycles. Figure 1 shows the landmark graph of the Sussman's anomaly slightly simplified for the sake of clarity.

Related Work

In this section a series of relevant previous works that have dealt with the landmark graph will be discussed.

Partitioning using Disjunctive Landmarks

The first proposed way of using landmarks was partitioning the problem into several smaller ones (Hoffmann, Porteous, and Sebastia 2004). This has the advantage of potentially obtaining an exponential gain by reducing the depth of the

problem. It works by taking the leaves (landmarks not preceded by other unachieved landmarks) of an acyclic landmark graph and turning them into a disjunctive set of goals. When this goal is achieved, the landmark graph is updated and the search begins from the last state with a new subjunctive goal until all landmarks have been achieved.

Although this achieves important speedups in some cases, it does not take into account interactions between goals and is forced to eliminate cycles from the landmark graph. Besides, the total order is guessed in a rather random way, as the search will almost always achieve the closest landmark even if others belonging to the goal must come first.

Landmark Layering

A more elaborated way of partitioning the problem is computing layers of conjunctive landmarks, as done in the planning system STELLA (Sebastia, Onaindia, and Marzal 2006). In this case the motivation is the same, but mutexes are taken into account along with the orderings to build sets of conjunctive goals. Basically layers are built delaying landmarks that are mutex with other landmarks depending on whether their causal preconditions have been achieved or not. Besides, cycles are dealt with by breaking them and assuming that one landmark must be achieved twice, guessing which one and at which time through some intuitions.

This method is more informed, but it is not without disadvantages. First, its computation is substantially more complex, to the point that in some instances the planner may time out even before beginning the search. Besides, it is an ad hoc approach based on a series of intuitions that make its generalization complex.

Temporal Landmarks

With the extensions that appeared in PDDL2.1 (Fox and Long 2003) and later versions time can also be characterized in a planning task. This lead to research on the discovery of temporal landmarks (L. Sebastia 2007). In this case a temporal planning graph is built after computing the regular STRIPS landmarks and new temporal landmarks are found for a given horizon. The most interesting point is that in the same process both types of landmarks get ”activated” at some point taking into account mutexes and orders, which gives an intuition of when they may be needed for the first time and even allows to prove that there is no solution for a given horizon if some landmark could not be activated in time.

In this case the novel concept of landmarks being required at some time step regarding the problem and its possible solution plans is introduced. However, landmarks still appear as required only once, cycles and unsound orderings are ignored and the whole computation of the activation times depends on a given horizon that is chosen by hand and whose viability is unknown.

Landmarks in a CSP to prove Solvability

With the deadline constraints introduced in PDDL3.0 (Gerevini et al. 2009) a hard constraint on the horizon of the planning task can be imposed. A different way of using the landmarks was proposed to detect unsolvable problems due to these time constraints (Marzal, Sebastia, and Onaindia 2008). In this case the landmark graph was encoded as a CSP, in which the landmarks were the variables, the time steps where the landmarks might be needed for the first time the variables and the orderings, mutexes and deadlines the constraints. Then, if the CSP had no solution, the problem was deemed as unsolvable.

In this case a more general approach is taken. Even though the goal is proving unsolvability, probably the most interesting part is the new landmark graph obtained from a solution assignment.

A SAT Compilation of the Landmark Graph

The planning graph and the landmark graph are two conceptually close concepts. In fact, the planning graph is the core of the most promising landmark discovery methods so far (Keyder, Richter, and Helmert 2010). However, the relationship between the two has not been further analyzed. Landmarks are sets of propositions and actions, so they can be represented in the planning graph as well. The main difference is that landmarks do not contain information regarding time. When carrying landmarks over to the planning graph, it is clear that a landmark is a proposition or action that is true at at least one level out of all the possible levels in which that proposition or action may appear. Hence, the question of discovering at which levels they have to be true involves finding both when and how many times.

Another related concept are the constraints that constitute the edges in both graphs. These edges are causal relationships between the nodes of the graph that encode time constraints. Since time steps are not represented in the landmark graph, its edges are of a more general nature. However, they share most of their properties and can be exploited in similar ways. Following this intuition, it is interesting to analyze whether some of these commonalities allow applying techniques used in the planning graph to the landmark graph.

An interesting approach commonly employed in automated planning is the encoding of the planning graph as a SAT problem. This is done by planners like Blackbox (Kautz and Selman 1999), which use a SAT solver to find an assignment that corresponds to a valid solution plan. Despite being a relatively old concept, these planners still represent the state of the art in parallel-length optimal planning as they effectively take advantage of the techniques developed by the SAT community. The classical way of translating encodes every proposition and action at every level as a variable, and every constraint as a clause. Since the number of parallel steps that the optimal solution has is unknown, the initial number of levels is set to the minimum required for the goal propositions to appear without being mutex. Then, the planning graph is converted into a SAT problem and solved using a SAT solver. If no solution is found, the planning graph is extended by one level and the process is repeated until a solution is finally found or the planning graph levels off, also known as the ”ramp-up” method.

Inspired by this approach a SAT compilation of the landmark graph is proposed. In this case, the variables represent the landmarks being true at every time step, and the clauses their respective ordering constraints. Additionally

londex constraints (Chen, Xing, and Zhang 2007), which are a generalization of static binary mutexes, are added to the problem. This is because the landmark graph does not contain explicit information regarding mutual exclusion, as opposed to the planning graph. The previous computation of h^{max} from the initial state for every landmark is also required for two reasons: first, landmarks should not be encoded as variables for levels in which they can not appear, as there must be a minimum of parallel time steps before some propositions can be achieved; and second, to obtain a minimum horizon (number of levels) from which begin the method, which is the maximum h^{max} among all the landmarks.

The same method as with SAT-based planners is used. Once the landmark graph has been computed, a SAT compilation for a given number of time steps is computed. Then, a SAT solver is used to find a solution assignment, and, if none is found, the horizon is increased. An important difference is that in this case the "ramp-up" method is not complete in the sense that it is only the compilation and not the landmark graph which varies from level to level and hence there is no equivalent concept to the planning graph leveling off.

Clause Encoding

Clauses can be divided in three types: existential clauses, ordering clauses and londex clauses. In these clauses, *ini* represents the time step at which a given proposition can be true for the first time. Existential clauses represent the fact that the landmarks must be true at some time at least once. They have the following form:

- Existential clauses: Every landmark must be true at at least one time step ($a_{ini} \vee ... \vee a_n$)

In the particular case of propositions that are true in the initial state and goals these clauses are not necessary, as they always appear at least in the first and last level respectively. The variables that represent these landmarks must be introduced in the problem, though, as they may be necessary at different time steps. These propositions however can be used to simplify the problem by setting the variables whose value is known to true and applying unit propagation.

Ordering clauses are actually the edges of the landmark graph. There is a different clause per type of sound ordering:

- Natural orderings: *a* must be true at some time step before *b* is true ($a_{ini} \vee ... \vee a_{t-1} \vee \neg b_t$)
- Necessary orderings: Either *a* or *b* must be true at the time step before *b* is true ($a_{t-1} \vee b_{t-1} \vee \neg b_t$)
- Greedy-necessary orderings: Either *a* or *b* must be true at some time step before *b* is true ($a_{ini} \vee ... \vee a_{t-1} \vee b_{ini} \vee ... \vee b_{t-1} \vee \neg b_t$)

In this case a clause is needed for every edge and time step in which a proposition can appear as the supported proposition, unless it is not possible to create the clause. The latter can happen when the precondition propositions can not appear at the required time steps for being those lower than their h^{max} value, in which case the supported proposition is assigned the value of false. For example, if *a* is naturally ordered before *b*, for every time step in which *b* can appear a clause must be created; however, if $h^{max}(a) \geq t$ then *a* can not be true before b_t and so b_t gets automatically assigned the value of false.

Necessary orderings are worth of mention, as they can induce cycles in the landmark graph. In this case though cycles are not undesirable, as they are resolved implicitly when finding a solution assignment. This allows to find important structural information in the planning graph, such as loops or a producer-consumer relationship between propositions. For instance, in the aforementioned Blocksworld domain necessary orders allow to discover that *(arm-empty)* is required in every even time step, as it is necessarily ordered before every *(holding x - block)*.

Reasonable orders behave differently from the rest of the edge constraints. Reasonable orders are said to be unsound as they do not have to be respected in every solution plan. However, they can be considered sound when regression on the propositions is done. Reasonable orders hold among goal propositions (Koehler and Hoffmann 2000). This means that if *a* and *b* are goal propositions and *a* is reasonably ordered before *b*, the last time *a* is made true must come before the last time *b* is made true. When doing regression, this means that b must be supported first whenever both a and b are true if both must be true until the final state. In fact, this is true for every pair of reasonable ordered propositions whenever they are true in the same state. In terms of setting a constraint, this means that if *a* and *b* are true at the same time and *a* is reasonably ordered before *b*, at least *a* must be true in the previous time step, because if an action would have added either proposition it should have been *b*. Both propositions staying as true until the last level is represented by forcing them to be true after *t* in every level. They only case in which this is not true is when there exists an action that adds both propositions at the same time, but in that case current methods would not report a reasonable order between them (Richter and Westphal 2010). Their representation as a SAT clause in the encoding of a landmark graph with *n* time steps is the following:

- Reasonable orderings: If *a* and *b* are true at the same level, *a* must be true at the time step before that level ($a_{t-1} \vee \neg a_t \vee \neg b_t \vee \neg a_{t+i} \vee ... \vee \neg a_n \vee \neg b_{t+i} \vee ... \vee \neg b_n$)

Before defining londex clauses, some clarifications must be done. Londexes are a generalization of mutexes that introduce a relation of mutual exclusivity among several time steps. For example, *(holding a)* and *(holding b)* in the Blocksworld domain would form a londex of distance 2, as at least 2 actions are needed to achieve one of the propositions from a state in which the other is true. Seen as a constraint, this means that both propositions could not be true neither in the same time step nor in consecutive time steps. A fact that is not mentioned in the definition of londex is that londex are not necessarily symmetrical, in the sense that if *a* and *b* are mutex the distance to achieve *b* from a state in which *a* is true may not be the same as the distance to achieve *a* from a state in which *b* is true. For example, in an instance of the Sokoban domain with a single block and in which the grid is a n×n one with no obstacles, a proposition

that represents the block being at some corner of the grid and another one that represents the block being somewhere at the center are mutex. However, the minimum number of actions to move the block from the center to the corner is finite, whereas moving the block from the corner to the center is not possible at all (in which case the distance could be considered as infinite). The clauses derived from the londex must take into account this fact, so they are based on the distance between mutex propositions rather than in a single londex constraint:

- Distance between londexes: *a* cannot be true at a time step *t'* if *b* is true at *t* such that $t - distance(a, b) > t' \leq t$ $(\neg a_{t-(distance-1)} \vee \neg b_t) \wedge ... \wedge (\neg a_t \vee \neg b_t)$

Disjunctive and Conjuctive Sets of Landmarks and Action Landmarks

Landmark discovery methods are not limited to single proposition landmarks. Disjunctive and conjuctive sets of landmark propositions and action landmarks may also be found. Regarding action landmarks, it is easy to see that their preconditions and effects are proposition landmarks themselves. Therefore, actions can be easily encoded with an additional variable with constraints over those landmarks. These constraints can be represented by clauses equivalent to those used when encoding the planning graph as a SAT problem. Two differences exist: first, there are no action levels in the landmark graph, so we must assume that either the preconditions or the effects are true at the same time step than the action; second, preconditions and added propositions are landmarks, but deleted propositions are not, so the only case in which these constraint would be represented is when the action deletes a proposition that is a landmark itself and so appears in the landmark graph independently from the action. In this case a regular mutex clause would be used. These are the clauses if it is assumed that actions occur at the same time their effects become true:

- Existential action clauses: Every action landmark must be true at at least one time step $(a_{ini} \vee ... \vee a_n)$
- Precondition clauses: Precondition *p* must be true one time step before action *a* is true $(p_{t-1} \vee \neg a_t)$
- Add clauses: Added proposition *p* must be true at the same time step than action *a* whenever *a* is true $(p_t \vee \neg a_t)$
- Delete clauses: Deleted proposition *p* must be false if it is a landmark at the same time step than action *a* whenever *a* is true $(\neg p_t \vee \neg a_t)$

Landmark actions can introduce non-causal landmark propositions in the landmark graph, this is, propositions that are not ordered before any other landmark and are added only as a side effect in every solution plan. Although other approaches ignore these landmarks in this case they are not harmful and the additional londex constraints they introduce may prove useful to add additional restrictions to the SAT encoding.

The sets of conjunctive and disjunctive propositions can be represented with auxiliary variables. Existential, natural and greedy-necessary ordering clauses are built for these auxiliary variables, whereas necessary and reasonable orderings and londex clauses are built for the propositions that compose the set. Auxiliary variables and their composing propositions are represented in the following way:

- Disjunctive landmarks: At least one proposition p^i from those that compose the set *s* must be true whenever *s* is true $(p_t^0 \vee ... \vee p_t^n \vee \neg s_t)$
- Conjunctive landmarks: Every propositions p^i from those that compose the set *s* must be true whenever *s* is true $(p_t^0 \vee \neg s_t) \wedge ... \wedge (p_t^n \vee \neg s_t)$

Exploiting the Solution of the SAT Encoding

As described before, the SAT encoding is iteratively generated by increasing its horizon from an initial value. A SAT solver is used for every resulting subproblem until a solution assignment is found. First of all, it should be noted that the solution may not be unique. First, there may be several solutions for the same landmark graph encoding; second, if there is a solution for an encoding with a given number of levels *n*, there will be solutions for every encoding with a number of levels greater than *n*. This means that the obtained graph is not sound in the sense that the total order obtained does not have to be respected by every solution plan. Besides, in the case of disjunctive landmarks the propositions that appear as true are not necessarily those that support the disjunctive set for every solution plan.

On the other hand, the use of a Max-SAT solver that is able to minimize the number of true variables is encouraged in order to prevent unnecessary assignments to true. These unnecessary assignments to true may occur when there are variables whose values do not affect the satisfiability of the solution. Nevertheless, Max-SAT is harder than regular SAT and so not using it or settling with a local minimum on the number of true variables is in most cases enough to get a representative solution. Another possibility is computing the backbone of the SAT problem; this is more informative but it is also harder to compute and may leave some landmarks with no value at all if they can appear as true at different time steps in different solution assignments.

By solving the problem, a time-stamped landmark graph is obtained. This assignment can be exploited in several ways. The first relevant conclusion is that the number of levels of the graph is a lower bound on the parallel length of the original problem. When regular SAT-based planners that employ the ramp-up method are used this does not represent a great advantage, as their running time is dominated by proving that there is no solution at the level *n-1* when the optimal solution has *n* levels. However, for other approaches that are based on guesses over the minimal parallel length of the problem (Xing, Chen, and Zhang 2006; Rintanen 2010) this can be used to reduce the range of horizons considered.

Another advantage of the time-stamped landmark graph is that an intuition about the times steps at which a landmark may be necessary is obtained. This allows the construction of some sort of roadmap that can be used to guide the search. Closely related with this idea is the aforementioned partitioning of the problem by using layers of conjunctive land-

marks (Sebastia, Onaindia, and Marzal 2006), although in this case the layers can be built just by choosing the landmarks that appear as true at every particular level with no additional computation.

An important difference with the original graph is that here landmarks may appear as true several times. This can mean two things: first, a landmark must not only be achieved but also maybe kept as true across different levels; second, it is often the case that a landmark *l* appears as true at non-consecutive time steps t_i, t_j and there is another landmark *l'* mutex with *l* which appears as true at some level t' such that $t_i < t' < t_j$. If this can be proven for every solution plan this means that *l* must be achieved, deleted and the achieved again. In conjunction with landmark-based heuristics this would allow to obtain more informative values; in particular, admissible landmark-based heuristics would keep their admissibility while not being limited by the upper bound that h^+ imposes to admissible delete-effect relaxation heuristics.

An Example of Time-Stamped Landmark Graph

In Figure 1 a simplified landmark graph of the Sussman's anomaly is shown. Although the graph contains the most relevant propositions and orderings, it gives little insight on what the structure of the task may be like. In particular, the cycles induced by the necessary orders on *(arm-empty)* and which represent the concept of the arm as a common resource in Blocksworld are absent, apart from the lack of minimum number of levels. For this reason this problem has been chosen as an example of the proposed approach.

Table 1 represents the output of the encoding process. LAMA (Richter and Westphal 2010) was used to generate the landmark graph and domain transitions graphs were used for the londex computation. Changes were done in its algorithm, as LAMA does not compute necessary orders. Besides, cycles in the landmark graph were not removed. The table shows at which levels landmarks must be true to satisfy the constraints. The opposite is not true; a landmark does not have to be false when it appears as not required. That a landmark must be false may be useful in some cases, although this is trivially computable using the original constraints along with the solution.

This solution is a good example of how some of the shortcomings of the landmark graph can be overcome. First, the number of levels is the minimum required; second, trivial landmarks like *(arm-empty)* being required on every even level but the last one are detected; third, the positions at which landmarks appear as needed offer a great deal of information with regards to possible solutions. In this notable case, the optimal solution plan always respects the required landmarks at the exact times; the other way around, the only sequence of actions that would comply with the solution of the compilation of the landmarks graph is the optimal plan.

Experimentation

Although this work is centered around an alternative method of exploiting the information contained in the landmark graph as a departure point for novel approaches, an experimental part has been added for the sake of completeness.

Level:	0	1	2	3	4	5	6
(arm-empty)	x	-	x	-	x	-	-
(on a b)	-	-	-	-	-	-	x
(on b c)	-	-	-	-	x	x	x
(on c a)	x	-	-	-	-	-	-
(clear a)	-	x	x	x	x	-	-
(clear b)	x	-	x	-	-	x	-
(clear c)	x	-	x	x	-	-	-
(on-table a)	x	-	-	-	-	-	-
(on-table b)	x	-	-	-	-	-	-
(holding a)	-	-	-	-	-	x	-
(holding b)	-	-	-	x	-	-	-
(holding c)	-	x	-	-	-	-	-

Table 1: Output of the solution of the SAT problem generated from the landmark graph

In this experiments the landmark layering approach used by STELLA (described in subsection *Landmark Layering*) has been emulated in the following way:

- The landmark graph is compiled into a SAT problem.
- A solution for the resulting SAT encoding is found using the "ramp-up" method employed by conventional SAT solvers.
- The problem is partitioned by creating a series of ordered subgoals. Subgoals are sets of conjunctive landmarks that appear as needed in a given layer of the new landmark graph. These subgoals are ordered by the level they were extracted from.
- A base planner is used to solve the different subproblems. The initial state for every subproblem is the final state from the previous one. The final solution is the concatenation of the solution plans of all the subproblems.

The base planner is Fast-Downward (Helmert 2006) with greedy best-first search as the search algorithm, the FF heuristic and preferred operators with boosting enabled. Experiments have been done on a Dual Core Intel Pentium D at 3.40 Ghz running under Linux. The maximum available memory for the planner was set to 1GB, and the time limit was 1800 seconds. Only non-disjunctive propositional landmarks were used in the encoding of the landmark graph. No parameter setting was necessary.

Experiments were done using the same domains that were used when STELLA was compared to other planners: Blocksworld, Elevator, Freecell, Logistics, Depots, Driverlog, Satellite and Zenotravel from the second and the third international planning competitions. Mutexes were computed using the invariant discovery process of Fast-Downward. Since this method is unable to detect important mutexes in some domains, three domains were left out: Elevators, Depots and Freecell.

Regarding coverage, the results were exactly the same, that is, both approaches were able to solve the same instances. This is due to the fact that the base planner is able to solve all the instances and that the domains have no deadends, which may make the partitioning approach unable to

solve some problems due to its greediness. Therefore, we will focus on quality and number of expanded nodes to compare both approaches.

Table 2 shows the comparison between the base planner and the partitioning approach in terms of evaluated states and solution quality as length in the Blocksworld domain. Instances that were solved by both approaches expanding fewer than 100 were left out, as they are deemed to easy to contribute with relevant information. Results show that in many cases partitioning leads to an increase in quality. Instances in which the partitioning approach finds shorter plans usually require fewer nodes expansions for this approach as well. The opposite phenomenon can be observed too: when the base planner finds shorter plans the number of states is also smaller. In this case, the differences are greater, which explains why the average number of expanded states by the partitioning problem is considerably bigger.

Problem:	Base-S	Part-S	Base-Q	Part-Q
Prob-9-0	145	112	72	28
Prob-9-1	141	108	60	28
Prob-9-2	73	104	44	26
Prob-10-0	228	136	60	34
Prob-10-1	148	128	60	32
Prob-10-2	145	136	52	34
Prob-11-0	98	2994	54	72
Prob-11-1	327	628	104	64
Prob-11-2	150	3038	72	62
Prob-12-0	269	30836	74	125
Prob-12-1	131	248	102	52
Prob-13-0	678	284	108	68
Prob-13-1	391	674744	108	120
Prob-14-0	238	5968	88	140
Prob-14-1	268	390732	94	342
Prob-15-0	1504	11052	184	144
Prob-15-1	498	604	124	114
Prob-16-1	455	514	102	96
Prob-16-2	2006	29034	160	118
Prob-17-0	2424	190	280	48
Geometric Mean	299.91	1419.31	87.32	68.30

Table 2: Comparison between the base planner and the partitioning approach in terms of evaluated states (columns "Base-S" and "Part-S") and solution quality as plan length (columns "Base-Q" and "Part-Q").

In the other domains the results are not so conclusive. The geometric mean of the number of evaluated nodes and the plan length was computed for every domain and the ratio (the base planner mean divided by the partitioning approach mean) displayed. Table 3 shows these results. On average the number of expanded is greater, whereas quality remains the same. This is explained by two factors: first, the base planner is already very competitive and so there is little margin for improvement, and second the usage of preferred operators with the FF heuristic already guides the search towards unachieved landmarks, which overlaps with the usage of the landmark graph. A partitioning scheme with a planner based in other paradigms like LPG(Gerevini and Serina 2002) and VHPOP(Younes and Simmons 2003) would probably yield more positive results.

Problem:	Mean States	Mean Quality
Driverlog	0.89	1.06
Logistics	0.8	0.87
Satellite	0.45	1.01
Zenotravel	0.54	1

Table 3: Comparison between the base planner and the partitioning approach for the rest of the evaluated IPC-2 and IPC-3 domains

Conclusions and Future Work

In this work an alternative representation of the landmark graph has been proposed. We have discussed the shortcomings of the original landmark graph and analyzed previous related work that has exploited the information contained in it. The points in common between the planning graph and the landmark graph have been brought up and a parallelism between the SAT encoding of the constraints of both graphs has been presented. Finally, we have described the SAT encoding of the landmark graph and analyzed its characteristics.

We have also proposed several ways of exploiting the new landmark graph that can lead to future lines of research. Some of the information obtained after solving the SAT problem generated by the landmark graph, like the minimum number of levels, can be straightforwardly used with current techniques. The structure of the graph itself can also be exploited as it has been done in the experimentation section or in alternative ways, for example by using it as the seed for local search planners like LPG. Also the fact that landmarks can appear several times may lead to more accurate landmark-based heuristics for forward search planners both in satisfying and optimal search. Finally, generalizing the SAT encoding of the landmark graph for temporal and numeric domains by transforming it into a constraint satisfaction problem is also a promising continuation of this work.

Acknowledgments

This work has been partially supported by a FPI grant from the Spanish government associated to the MICINN project TIN2008-06701-C03-03.

References

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artif. Intell.* 90(1-2):281–300.

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In *ECAI*, 329–334.

Chen, Y.; Xing, Z.; and Zhang, W. 2007. Long-distance mutual exclusion for propositional planning. In *IJCAI*, 1840–1845.

Fox, M., and Long, D. 2003. Pddl2.1: An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)* 20:61–124.

Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs with action costs. In *AIPS*, 13–22. AAAI.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.* 173(5-6):619–668.

Helmert, M. 2006. The Fast Downward planning system. *J. Artif. Intell. Res. (JAIR)* 26:191–246.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *J. Artif. Intell. Res. (JAIR)* 22:215–278.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*, 1728–1733.

Kautz, H. A., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In *IJCAI*, 318–325. Morgan Kaufmann.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In *ECAI*, 335–340.

Koehler, J., and Hoffmann, J. 2000. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *J. Artif. Intell. Res. (JAIR)* 12:338–386.

L. Sebastia, E. Marzal, E. O. 2007. Extracting landmarks in temporal domains. In *International Conference on Artificial Intelligence (ICAI'07)*, volume 2, 520–525.

Marzal, E.; Sebastia, L.; and Onaindia, E. 2008. Detection of unsolvable temporal planning problems through the use of landmarks. In Ghallab, M.; Spyropoulos, C. D.; Fakotakis, N.; and Avouris, N. M., eds., *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, 919–920. IOS Press.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res. (JAIR)* 39:127–177.

Rintanen, J. 2010. Heuristics for planning with SAT. In Cohen, D., ed., *CP*, volume 6308 of *Lecture Notes in Computer Science*, 414–428. Springer.

Sebastia, L.; Onaindia, E.; and Marzal, E. 2006. Decomposition of planning problems. *AI Commun.* 19(1):49–81.

Xing, Z.; Chen, Y.; and Zhang, W. 2006. Optimal strips planning by maximum satisfiability and accumulative learning. In Long, D.; Smith, S. F.; Borrajo, D.; and McCluskey, L., eds., *ICAPS*, 442–446. AAAI.

Younes, H. L. S., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *J. Artif. Intell. Res. (JAIR)* 20:405–430.