BRT: Biased Rapidly-exploring Tree

Vidal Alcázar
Universidad Carlos III de Madrid
Avenida de la Universidad, 30
28911 Leganés, Spain

Manuela Veloso
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA

Abstract

This paper describes the planner BRT (Biased Rapidly-exploring Tree). This planner is basically a Rapidly-exploring Random Tree (RRT) adapted to automated planning that employs Fast-Downward as the base planner. The novelty in this case is that it does not sample the search space in a random way; rather, it estimates which propositions are more likely to be achieved along some solution plan and uses that estimation (called bias) in order to sample more relevant intermediates states. The bias is computed using a message-passing algorithm on the planning graph with landmarks as support.

Overview

Landmarks are disjunctive sets of propositions or actions of which at least one component must be achieved or executed at least once in every solution plan to a problem (Richter, Helmert, and Westphal 2008). However, computing the complete set of landmarks or even proving that a proposition or action is indeed a landmark is PSPACE-complete (Hoffmann, Porteous, and Sebastia 2004).

A notable similarity exists between landmarks in automated planning and backbone variables (Kilby et al. 2005) in CSPs. A backbone variable is defined as a variable that takes the same value in every solution for a given problem. Again, determining which are the backbone variables in a problem is intractable in general. In order to overcome this, approximate methods that compute an estimation of the probability that a variable has of taking a given value - also known as bias - have been proposed. In particular, message-passing algorithms appear to be well suited to this kind of probabilistic environment.

Biases can be seen in automated planning as the probability of a proposition or an action being respectively achieved or executed in a solution plan. Extending the links between these two cases, just like backbones variables are variables with a bias of 1 for a given value, landmarks could also be defined as having also a bias of 1. Because of the characteristics of planning problems, though, a straightforward correlation between variable bias and proposition/action bias does not exist. In particular, the fact that it is relevant at which time step variables and actions must be achieved or executed invalidates these assumptions. To take into account this fact, a planning graph (Blum and Furst 1997) is used so the bias is computed for every variable and action at each time step in which they can appear.

The constraints encoded by the planning graph alone may not suffice to find a good bias. To solve this, landmarks are added to the planning graph as sources of probability. This requires first having an estimation of where landmarks should appear in the planning graph, which is obtained by turning the landmark graph into a SAT problem that encodes time steps and using a SAT solver to get a possible assignment.

The general process is as follows:

- First, the landmark graph is encoded as a SAT problem and an estimation of when the landmarks are needed is obtained.
- Second, a planning graph enriched with the landmarks at the aforementioned estimated positions is generated and a message-passing algorithm is used to compute the bias - in this case Expectation Maximization Survey Propagation (Hsu and McIlraith 2009).
- Finally, biases are used to introduce probabilities in the sampling process of the RRT.

A SAT Compilation of the Landmark Graph

Current methods for computing landmarks are able not only to find landmarks, but also to establish partial orders between them. These orderings connect the landmarks forming a directed graph, the landmark graph. Figure 1 shows the landmark graph of the Sussman's anomaly slightly simplified for the sake of clarity. There are several ordering relationships between landmarks propositions (Richter, Helmert, and Westphal 2008):

- 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

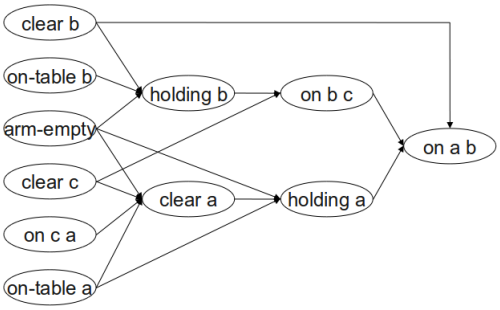

Figure 1: Simplified landmark graph of the Sussman's anomaly.

plan must delete *b* on the way to *a*, and re-achieve *b* after or at the same time as *a*

This representation does not take into account that a landmark may be needed at several time steps and does not provide an insight of the total order of the landmarks, as it only considers relative orders between them. Besides, it usually underestimates the minimum parallel length of the solutions to the problem. Figure 1 illustrates these facts: *(arm-empty)* should appear at every even level as opposed to only at level 0; the number of propositional layers is four, meaning that in theory a solution plan with only three actions may be possible, whereas the shortest parallel plan has 6 actions; *(holding a)* and *(holding b)* appear at consecutive levels, despite needing 2 actions to achieve either proposition from a state in which the other is true,... An earlier work that tried to group landmarks into layers in order to partition the problem (Sebastia, Onaindia, and Marzal 2006) partially addressed these issues, although not in an explicit way and with the inconvenience of being computationally expensive in some planning domains.

To solve this, a SAT compilation of the landmark graph is proposed. It is inspired by the SAT-compilation process of optimal makespan planners, in which propositions and actions are represented by a different variable at every time step. In this case the landmark propositions at the different levels are variables, and the clauses represent the different constraints that exist between them. Constraints are either the orderings that define the landmark graph or relations of binary mutual exclusivity. In particular, long distance mutexes (Chen, Xing, and Zhang 2007), which are a generalization of regular mutexes, will be used. The different types of clauses are the following (*ini* represents the time step at which a given proposition can be true for the first time):

- Existential clauses: Every landmark must be true at at least one time step ($a_{ini} \vee ... \vee a_n$)
- Natural orderings: *a* must be true at some time step before *b* is true ($a_{ini} \vee ... \vee a_{t-1} \cup \neg b_t$)
- 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 ($a_{ini} \vee ... \vee a_{t-1} \vee b_{ini} \vee ... \vee b_{t-1} \vee \neg b_t$)
- Reasonable orderings: If *a* and *b* are true at the same level, *a* must be true at the time step before that level ($a_{t-1} \vee \neg a_t \vee \neg b_t$)
- Distance between londexes: *a* cannot be true at a time step *t'* if *b* is true at *t* such that $t - distance(a, b) > t' \leq t$ $(\neg a_{t-(distance-1)} \vee \neg b_t) \wedge ... \wedge (\neg a_t \vee \neg b_t)$

The resulting encoding is solved using the "ramp-up" method most SAT-based planners employ: compile the problem with a given maximum length, try to find a solution using a SAT solver and, if no solution is found, repeat the process incrementing the maximum length. The solution may not be unique, so the final position of the landmarks is not sound and should be used only as an estimate.

By solving the problem, a time-stamped landmark graph is obtained. The first consequence is that the depth of the graph is a lower bound on the parallel length of the original problem. The second is that landmarks may appear as true several times, meaning that they must be true at different time steps. The third is that an intuition about the time steps at which a landmark may be necessary is obtained. It should be noted that the total order obtained does not have to be respected by every solution plan.

Table 1 represents the output of this process in the Sussman's anomaly. 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.

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

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.

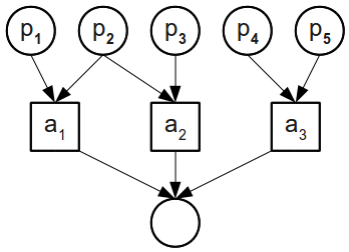

Figure 2: An example of bias computation. In this case the positive bias of p_2 is higher than the bias of the other propositions, as it appears as true in a higher number of solutions.

Estimating the Bias of Propositions and Actions

Message-passing compute the marginal distribution on the set of variables by relying on the structure of a factor graph, a bipartite graph whose nodes are either variables or constraints. Message-passing algorithms send values along the edges of the graph representing the influence between variables given the constraints that affect them. The method used in this work is Estimation-Maximization Survey Propagation (EMSP) with a global update rule (Hsu and McIlraith 2009). EMSP is a set of update rules derived using the Estimation Maximization framework for maximum-likelihood parameter estimation that has proved to be effective to solve random SAT problems. Survey propagation is essentially a variation of belief propagation in which the bias, instead of being split as positive or negative, accepts a third value, the "joker" or "don't care" one. This value represents the probability of the value of the variable not being critical for the satisfiability of the solution; this is, whether a solution remains valid when changing the value of the variable. On the other hand, the Estimation-Maximization framework has the advantage of guaranteeing convergence even in graphs with loops, unlike regular message-passing algorithms. This way, EMSP initializes the bias of the variables with random values and iterates by doing a two-step process: first, every variable sends its bias to the constraints it appears in, allowing the computation of the probability of those constraints to be satisfied. Second, every variable updates its bias based on the previously computed probability.

Figure 2 shows an example of what biases mean in a SAT problem. In this case, there are three actions a_1, a_2, a_3 that achieve a goal and five propositions p_1, p_2, p_3, p_4, p_5 that are the preconditions of the actions as represented by the arrows. Only one action can be executed, which means that they are mutex between them. Every action and proposition besides the goal is a variable. The clauses are $(a_1 \vee a_2 \vee a_3)$, $(\neg a_i \vee \neg a_j)$ for each $i \neq j$ and $(p_i \vee \neg a_j)$ for each $p_i \in pre(a_j)$. If EMSP is used, the positive bias of p_2 is 0.5 whereas the bias of the rest of the propositions is around 0.3 This is because out of all the possible solution assignments to the problem, p_2 appears as true in more cases. Also this example shows why choosing Survey Propagation instead of regular Belief Propagation may be useful: let's suppose a_1 was chosen as the supporting variable for the clause $(a_1 \vee a_2 \vee a_3)$. In that case the values of p_1, p_2, a_2, a_3 are enforced by the constraints. However, the value of the rest of the propositions does not matter - they could be true or false and the assignment would still satisfy the constraints. Survey Propagation captures this notion, which allows to compute the positive and negative biases without being affected by situations like the one described.

This planning graph can be easily seen as a factor graph in which the factor vertices are the constraints that are represented by the edges and mutexes of the planning graph. A reasonable formulation of the planning graph as a factor graph is its compilation to a SAT problem, as done by SAT-based planners like Blackbox (Kautz and Selman 1999). In this case, the variables represent the different propositions and actions at different time steps, and the clauses are the constraints between them. This choice implies that the estimated bias would be about actions and propositions *at given time steps*. This is in opposition to the concept of landmark, which states that they must be true along every solution plan without any particularization of when.

To improve the inference process landmarks are inserted in the positions obtained from the solution of the SAT encoding of the landmark graph and given a bias of 1. Furthermore, the number of levels of the solution obtained from the SAT compilation of the landmark graph is a sound lower bound on the parallel length of any possible solution plan, so the planning graph should have at least that many levels. Algorithm 1 gives an outline of the whole process done when computing the bias of actions and propositions.

Algorithm 1: Computation of biases enhanced by landmarks

```
Data: Current problem P = (S, A, I, G)
Result: Survey survey
begin
    LandmarkGraph ⟵ computeLandmarks(P)
    MinimumLevel ⟵ computeHmax(P)
    LandmarkSAT ⟵ NULL
    while notLandmarkSAT do
        LandmarkEncoding ⟵
        compileLandmarks(LandmarksGraph,
        MinimumLevel)
        LandmarkSAT ⟵
        solve(LandmarkEncoding)
        MinimumLevel ⟵ MinimumLevel + 1
    PlanningSAT ⟵
    generateCNF(P, MinimumLevel − 1)
    PlanningSAT ⟵
    insertLandmarks(PlanningSAT, LandmarkSAT)
    survey ⟵ computeSurvey(PlanningSAT)
    return survey
end
```

Biased RRT

One of the possible uses of the biases is sampling the search space. The goal is to find intermediate states that may be

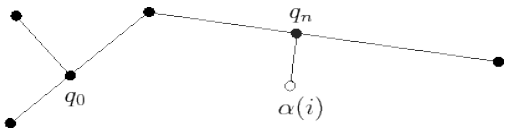

Figure 3: Search phase of the RRT. The local search expands towards the rightmost sampled state with a limit. When the limit is reached, q_n is added to the tree.

relevant to the search. Doing this the problem can be partitioned into several smaller problems, which greatly decreases the complexity of the task. An incremental algorithm can be used to try to reach those states and build a path to the goal state. A baseline planner is used to solve the different subproblems. In this case, we will use a Rapidly-Exploring Random Tree (RRT) (Kuffner and LaValle 2000) adapted to automated planning following some of the ideas introduced by RRT-Plan (Burfoot, Pineau, and Dudek 2006). Fast-Downward with the FF heuristic, preferred operators and greedy best first search with lazy evaluation will be the base planner.

RRTs try to find a path to the goal by randomly sampling the search state and trying to join the closest node of the tree to the sampled state. The algorithm works as follows: first, a state that is not inside an obstacle is sampled; then, the closest node of the tree is found using a measure of distance. After having found the closest node, a baseline planner is called with a limit on time or length that tries to reach the sampled state. When it is reached or the planner surpasses the imposed limit, the last expanded state is added as a new node of the tree with an edge connecting it to the node that served as initial state for the subproblem. In single-query RRTs after expanding towards a sampled state often a new search is invoked trying to join the newly created node with the goal state. Another approach used to speed up convergence is, instead of sampling, with a probability *p* the closest node to the goal is chosen and expanded towards it. Figure 3 shows how the tree is built towards the goal while sampling at the same time.

Mutexes are used when sampling to avoid selecting unreachable steps and a reachability analysis from the sampled state is done to avoid dead-end states. Besides, sampling with a random probability may lead to exploring areas of the search space which are not relevant to the solution of the problem. To prevent this from happening, the use of biases is proposed. Sampling is thus a several steps process: first, a random proposition level of the planning graph is chosen, as biases differ from level to level. Second, a variable from the SAS+ formulation of the problem is randomly chosen. After that, a proposition belonging to the invariant is picked using roulette wheel selection with the bias as the probability of selection. When a proposition is chosen all the propositions belonging to other variables that are mutex with the chosen one are discarded. This process is repeated until a proposition from each variable has been selected. If a variable is selected and has no valid propositions the sampled state is unreachable due to mutexes and the process is restarted from scratch.

Acknowledgments

This work has been developed when Vidal Alcázar was visiting Manuela Veloso at Carnegie Mellon University 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.

Burfoot, D.; Pineau, J.; and Dudek, G. 2006. RRT-Plan: A randomized algorithm for STRIPS planning. In *ICAPS*, 362–365. AAAI.

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

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

Hsu, E. I., and McIlraith, S. A. 2009. Varsat: Integrating novel probabilistic inference techniques with DPLL search. In *SAT*, volume 5584 of *Lecture Notes in Computer Science*, 377–390. Springer.

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

Kilby, P.; Slaney, J. K.; Thiébaux, S.; and Walsh, T. 2005. Backbones and backdoors in satisfiability. In *AAAI*, 1368–1373. AAAI Press / The MIT Press.

Kuffner, J. J., and LaValle, S. M. 2000. RRT-Connect: An efficient approach to single-query path planning. In *ICRA*, 995–1001. IEEE.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *AAAI*, 975–982. AAAI Press.

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