RRT-Plan: a Randomized Algorithm for STRIPS Planning

Daniel Burfoot and **Joelle Pineau** and **Gregory Dudek**
Centre for Intelligent Machines
McGill University, Montreal, Canada
{burfoot, dudek}@cim.mcgill.ca, jpineau@cs.mcgill.ca

Abstract

We propose a randomized STRIPS planning algorithm called RRT-Plan. This planner is inspired by the idea of Rapidly exploring Random Trees, a concept originally designed for use in continuous path planning problems. Issues that arise in the conversion of RRTs from continuous to discrete spaces are discussed, and several additional mechanisms are proposed to improve performance. Our experimental results indicate that RRT-Plan is competitive with the state of the art in STRIPS planning.

Introduction

In this paper we propose a new approach to planning in discrete domains, and demonstrate the approach in the context of STRIPS planning. The planning problem is formulated as a search through a large state space. To effectively explore this space, we employ a strategy based on randomization.

Heuristic planners such as HSP (Bonet & Geffner 1999) and FF (Hoffmann & Nebel 2001) define a state-specific evaluation function to guide search through the state space. While this approach is often quite successful, in many situations the guidance provided by the heuristic function is incorrect or misleading. In particular, the topology of the search space will often exhibit large plateaus or local minima, which must be exhaustively searched. To avoid this, our algorithm combines large scale stochastic exploration with limited local searches. We argue that this provides a large degree of robustness to problems where heuristic evaluations are imperfect.

The notion of Rapidly-exploring Random Trees (RRTs) was introduced by (LaValle 1998), and applied to robot path planning. RRTs are known to be particularly useful in path planning domains where the state space is very large, but solutions are not scarce. With the notable exception of (Morgan & Branicky 2004), RRTs have not been extended to discrete-space planners. These authors considered only the general case of discrete search, and did not leverage any of the ideas or techniques that have been developed for STRIPS planning.

The primary contribution of this paper is a STRIPS planning algorithm which is inspired by RRT-style randomized exploration. Techniques are presented to deal with several difficulties that arise when attempting to translate RRT ideas from the continuous case to the discrete case. A new idea called goal subset locking is introduced which helps to avoid a common mistake made by the relaxed plan heuristic. Finally, results are shown which validate the performance of RRT-Plan compared to the state-of-the-art planners FF (on whose technology we rely significantly), and LPG (Gerevini & Serina 2002).

In the following we assume the reader is familiar with the STRIPS planning formalism (Fikes & Nilsson 1971) and the concept of relaxed plan heuristic (h^+) search introduced by (Bonet & Geffner 1999). We also make reference to the algorithm known as Enforced Hill Climbing developed by (Hoffmann & Nebel 2001)

Rapidly-exploring Random Trees

Rapidly-exploring Random Trees (RRTs) were first introduced by (LaValle 1998) to solve multidimensional path planning problems. An RRT is a tree composed of nodes which represent positions in the search space. The RRT growth process causes it to rapidly expand throughout the space. Initially the tree has only one node located at the starting state. Then we choose a point q_{rand} at random in the state space. We find q_{rand}'s nearest neighbor in the tree, that state is denoted q_{near}. Next we try to connect q_{rand} and q_{near} using a simple local planner, which essentially just moves in a straight line from one point to the other until it encounters an obstacle. If the local planner succeeds in reaching q_{rand}, it is added as a new node in the tree. Typically the search is only allowed to proceed for some distance ϵ, in which case the location reached (called q_{new}) is added instead. This process is illustrated in Figure 1. The distance ϵ is a parameter that must be set. It can be shown that, given enough time, the RRT will grow to uniformly cover the state space (LaValle 1998).

(Kuffner & LaValle 2000) extended the basic RRT concepts to create an algorithm called RRT-Connect. This algorithm works by growing an RRT from the start state, and after every expansion phase attempting to connect to the goal. Because of the uniform sampling property of the tree, the algorithm is probabilistically complete.

RRTs are intended to find solutions in large problems where optimality is not critical. A good example of the

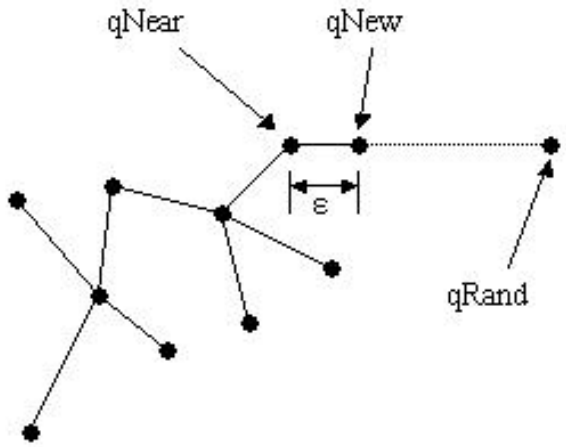

Figure 1: RRT Expansion process. q_{rand} is a randomly chosen point, q_{near} is its nearest neighbor in the tree. The local planner is only allowed to progress a distance ϵ, after which a new point is added and the process repeats.

kinds of problems that can be solved easily by RRTs is that of a humanoid robot picking up an object (Kuffner *et al.* 2003). The robot has many degrees of freedom, creating a high-dimensional search space; however, there are also many valid solutions.

In many STRIPS planning problems, an analogous situation occurs: the state space is very large, but solutions are not scarce. This motivates our attempt to translate the RRT concept into discrete domains.

The RRT-Plan Algorithm

RRT-Plan is a STRIPS algorithm based on RRT concepts. At a high level, our RRT-Plan contains the same steps as the RRT-Connect algorithm described above. An outline is provided in Figure 2. The basic idea is to randomly expand a tree over the state space until a node is found that is sufficiently close to the goal that it can be achieved with a local search.

There are several obstacles to extending the concepts of RRTs to the discrete planning case. First, RRTs require the ability to randomly sample from the state space. Second, given a random state, RRTs require the ability to find its Nearest Neighbor in the tree. Finally, a deterministic planner must be invoked to try to connect nodes to random states and to the goal. We now discuss each step of our algorithm in detail.

Select random state. The first step relies on the ability to sample points randomly from the space. While it is not difficult to sample randomly, it is very hard to sample the reachable space uniformly. This is because determining if any given state should be assigned a non-zero probability (i.e. determining if it is reachable) is equivalent to solving the planning problem itself.

Because of this difficulty, RRT-Plan generates target states q_{rand} by taking random subsets from the goal atoms.

- **Select random goal subset RGS**
- **Find "Nearest Neighbor" qNear to RGS**
- **Invoke planner to connect qNear to RGS**
- **If state qNew is found that satisfies RGS,**
  - Add qNew to tree as child of qNear
  - Calculate atom costs for qNew
  - Attempt to connect qNew to final goal

Figure 2: One iteration of the RRT-Plan algorithm.

This approach has several advantages. It is easy to compute. If the problem is solvable, then every goal subset must itself be reachable. Finally, it tends to bias the search towards the goal. In the following we abbreviate "random goal subset" as RGS.

Some domains have natural goal orderings. To take these into account, we use the goal agenda heuristic of (Koehler & Hoffmann 2000) when generating an RGS. The ordering would eventually be found through trial and error, but using the goal agenda speeds up the process.

Find Nearest Neighbor. The second step in the expansion process requires finding the node in the tree that is closest to the random target. Unfortunately finding the precise distance between two states is again equivalent to the planning problem itself.

We adopt the HSP technique h^+_{add} to estimate distances between states. For a given state, this provides an estimate of the cost to achieve every atom in the problem. Thus the costs, once calculated, can be reused for every nearest neighbor query.

Invoke sub-planner to connect q_{near} to q_{rand}. RRT-Plan requires a sub-planner at two steps - first to connect the nearest neighbor q_{near} to the target q_{rand}, and second to connect the new node q_{new} to the goal. We apply a limited version of FF as a local planner. Only the fast Enforced Hill Climbing phase is used, and a node expansion limit is enforced, after which the search is abandoned. These restrictions are analogous to the ϵ parameter in continuous RRTs.

Add q_{new} to tree. We maintain a simple tree structure in memory. When a goal subset q_{rand} is reached from q_{near}, a new node is added to the tree as a child of q_{near}, and the actions required to reach q_{rand} are stored. If the full goal is reached from a node in the tree, we can construct a full solution by following the path from the root to the node and then to the goal (note that soundness of the full algorithm is guaranteed by the soundness of the local planner). If the sub-planner fails to connect to the target, no new node is added to the tree.

Attempt to connect q_{new} to goal. Again, we use the Enforced Hill Climbing phase of FF, with bounded node expansion. In contrast to the previous connection attempt, in this case the best resulting node is kept and added to the tree. This is because it might be very close to the goal even if the search does not succeed.

Importantly, this allows for the node expansion limit to be effectively bypassed. Consider the following scenario: a new RRT node is created, and then an attempt is made to reach the goal. This attempt fails because of the node expansion limit, but would have succeeded if it had been allowed to continue. Because the resulting state is also added as a new node, it can be selected as nearest neighbor on the next iteration, and the search can be continued from the point it was halted due to the expansion limit.

Goal Subset Locking. Several of the problems encountered by h^+ planners are caused by the fact that the relaxed plan length heuristic does not penalize the deletion of goal atoms. If a goal atom can be achieved in an "easy" way through the deletion of an already asserted goal atom and in a "hard" way (in which other goal atoms are not deleted), the heuristic value is low, corresponding to the "easy" way. Furthermore, and perhaps more problematically, states which are closer to or further from the solution along the hard path are not accorded correspondingly better or worse values.

To avoid such situtations, when a RGS is achieved in the connection phase of RRT-Plan, the atoms of the RGS are *locked* so that any future action which deletes them is not considered. Additionally, any goal atoms which are locked in a parent node are also locked in its children. Importantly, this restriction is taken into account when calculating the atom cost estimates for the node. States for which the final goal is accorded an infinite heuristic value are discarded. We define $h^+_{gsl}(s, gs)$ to be the length of the relaxed plan from s to the goal where actions which delete atoms in goal subset gs are disallowed. It is clear that this modification can never decrease the heuristic evaluation of a state. Also, it is admissible with respect to the restricted planning problem.

Adapting Search Parameters. We can extract useful information from the growth (or failure to grow) of the tree. This information is used to adapt the parameters of the subplanner. Roughly, statistics are kept regarding which goal atoms are easy to achieve and which are difficult. When searching for difficult goal atoms, the local planner is allowed to expand a greater number of states. Similarly, we keep track of which nodes in the RRT look like they might be dead ends, and discriminate against those nodes in Nearest Neighbor queries. These techniques are described at greater length in (Burfoot, Pineau, & Dudek 2006).

Experimental Results

To validate RRT-Plan, we compared its performance with that of state-of-the-art planners FF and LPG-td 1.0 (speed settings) on problems[1] from the planning competitions from 1998, 2000, and 2002 (McDermott 2000; Bacchus 2001; Long & Fox 2003). We also used the STRIPS version of the Pipesworld domain from the 2004 competition.

In addition we used a simple new domain called Push-Block, to illustrate a case in which RRT-Plan's techniques are a substantial improvement. This is simply a 20x20 grid of positions which can be occupied by blocks. The blocks can be pushed left, right, up or down, but not into a location which already contains another block. The goal is simply a set of locations to which we must move blocks. We generated 20 domains, starting with one block/goal atom and moving up to 20.

[1] In Blocks-World only the first 35 problems from 2000 were used; FreeCell problems come from both 2000 and 2002 competitions; the Logistics problems came from Track1 of 2000 including the Additional set; Typed versions of domains were used when available.

Table 1: Performance of FF, RRT-Plan, and LPG on various domains. Entries list the number of problems the planner could not solve within five minutes of CPU time.

Domain	FF	RRT-Plan	LPG
Blocks World (35)	3	1	0
Driverlog (20)	5	0	0
Depot (22)	3	0	0
Freecell (80)	8	10	70
Logistics (63)	0	0	0
MPrime (35)	3	3	0
Mystery (30)	14	13	12
Pipesworld (50)	15	8	9
Rovers (20)	0	0	0
Satellite (20)	0	0	0
Push-Block (20)	15	0	19

Table 1 shows the number of problems solved by the three planners on the test suite. These statistics were generated by allowing the planners to run for up to five minutes and recording the time to completion. Experiments were performed on a 3GHz Pentium 4 Linux machine with 2GB of RAM. Looking at the table, we see that RRT-Plan outperforms FF in several domains and is worse in only one, FreeCell.

Looking at the results, we distinguish three cases. First, there are domains which FF solves easily (Logistics, Rovers, Satellite). Here RRT-Plan succeeds easily as well, on the first goal connection attempt. Second, there are problems which have very few goal atoms (Mystery, MPrime, FreeCell). This hinders the RGS scheme employed by RRT-Plan. However, the algorithm will realize this and effectively channel all computational resources to the local planner, so it again becomes equivalent to FF. In several other domains (DriverLog, Depots, Pipesworld, Push-Block), randomization yields significant performance benefits. Push-Block is an extreme example of a domain in which goal atoms can be achieved quickly, but only at the price of deleting other goal atoms. This means the h^+ heuristic gives egregiously bad estimates. RRT-Plan can handle this through randomization and goal subset locking. More in-depth analysis is given in (Burfoot, Pineau, & Dudek 2006).

The goal of RRT-Plan is to find solutions to difficult problems. We were prepared to accept suboptimal plan lengths, and expected RRT-Plan's solutions to be uniformly worse than those of the other planners. Surprisingly, this was not always the case. Figure 3 shows the number of plans generated with less than a given number of actions. The curves for the three planners are nearly identical, indicating that RRT techniques do not significantly worsen plan length.

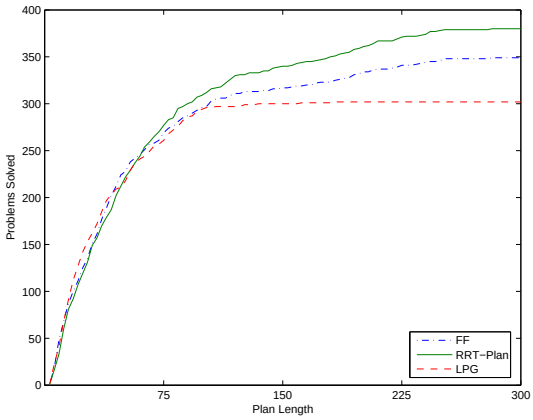

Figure 3: Plan length comparison for FF, LPG, and RRT-Plan.

Discussion

This paper presents a randomized algorithm for STRIPs planning. In general, one of the most important advantages of randomized over deterministic algorithms is that they avoid systematic errors. Heuristic evaluation functions often report misleading information, which can lead to large plateaus or local minima. If the search is guided only by the heuristic, the planner is required to exhaustively search such regions. RRT-Plan avoids this pitfall by using randomization for large scale exploration and limiting local searches. An important side benefit is that large open lists are not needed, reducing memory consumption. Note also that the downside of the Enforced Hill Climbing phase of FF - that it can fail if it reaches a dead end - does not particularly concern RRT-Plan.

One critical difference between continuous and discrete RRTs regards the volume of the "ϵ-ball", the region from which the goal can be achieved by the sub-planner. In continuous planning, the volume of the ϵ-ball is usually just one or two orders of magnitude smaller than the entire configuration space. By contrast in discrete planning, the ϵ-ball can be exponentially smaller than the state space, depending on the heuristic.

Instead of attempting to choose states randomly from the space, RRT-Plan instead randomly samples from goal subsets. As the number of achieved goal subsets in the tree grows, it moves closer and closer to the complete goal.

While RRT-style exploration is at the heart of our approach, several additional techniques are required. Most prominently is the idea of goal subset locking. By pointing the h^+ heuristic at a goal subset and preventing it from deleting already achieved goals, the topology of the search landscape is simplified. In some cases local minima and plateaus are removed. Also, as the tree grows certain goal atoms are identified as problematic, and more effort is devoted to achieving them.

A future objective is bi-directional RRT-Plan, in which another tree is grown from the goal and expands in the regression space of the problem. We also plan to add support for tasks involving numeric constraints.

Acknowledgments

We would like to thank the authors of FF and HSP for making the code to their planners available online.

References

Bacchus, F. 2001. The AIPS '00 planning competition. *AI Magazine* 22(3):47–56.

Bonet, B., and Geffner, H. 1999. Planning as heuristic search: New results. In Biundo, S., and Fox, M., eds., *Proc. 5th European Conf. on Planning*, 359–371. Durham, UK: Springer: Lecture Notes on Computer Science.

Burfoot, D.; Pineau, J.; and Dudek, G. 2006. A STRIPS algorithm based on Rapidly-exploring Random Trees. TR-CIM 06-02, Center for Intelligent Machines, McGill University.

Fikes, R., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. In *IJCAI*, 608–620.

Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs with action costs. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *AIPS*, 13–22. AAAI.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)* 14:253–302.

Koehler, J., and Hoffmann, J. 2000. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *Journal of Artificial Intelligence Research* 12:338–386.

Kuffner, J. J., and LaValle, S. M. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 995–1001.

Kuffner, J.; Nishiwaki, K.; Kagami, S.; Inaba, M.; and Inoue, H. 2003. Motion planning for humanoid robots. In *Proc. 11th Intl Symp. of Robotics Research (ISRR 2003)*.

LaValle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *J. Artif. Intell. Res. (JAIR)* 20:1–59.

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

Morgan, S., and Branicky, M. S. 2004. Sampling-based planning for discrete spaces. In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems*.