

On Heavy-tailed Runtimes and Restarts in Rapidly-exploring Random Trees

Nathan A. Wedge and Michael S. Branicky

Department of Electrical Engineering and Computer Science
Case Western Reserve University
10900 Euclid Avenue, Cleveland, OH 44106
nathan.wedge@case.edu, mb@case.edu

Abstract

Randomized, sampling-based planning has a long history of success, and although the benefits associated with this use of randomization are widely-recognized, its costs are not well-understood. We examine a variety of problem instances solved with the Rapidly-exploring Random Tree algorithm, demonstrating that heavy-tailed runtime distributions are both common and potentially exploitable. We show that runtime mean and variability can be reduced simultaneously by a straightforward strategy such as restarts and that such a strategy can apply broadly across sets of queries. Our experimental results indicate that several-fold improvements can be achieved in the mean and variance for restrictive problem environments.

Introduction

Randomized, sampling-based approaches to problem-solving have become commonplace in planning. They bring noteworthy advantages over existing deterministic methods, perhaps the most significant of which is their suitability for high-dimensional problems. However, their use carries with it a significant drawback: the presence of random sampling implies that the runtime of the algorithm is also random. Although common algorithms such as the Rapidly-exploring Random Tree (RRT – LaValle 1998; LaValle and Kuffner 2001) and the Probabilistic Roadmap (PRM – Kavraki et al. 1995) are *probabilistically complete* (guaranteed to terminate with probability approaching one), it is difficult to theoretically characterize the resulting runtime distribution. Additionally, runtimes may be highly variable, making use of this class of algorithms problematic in real-time or human-interactive applications.

The lack of predictability in solution quality and runtime has motivated a pursuit of deterministic and quasi-random versions of sampling-based planning algorithms (Branicky et al. 2001; LaValle, Branicky, and Lindemann 2004; Lindemann and LaValle 2004b). While promising, these deterministic versions have not yet demonstrated that they

can provide definitive advantages over their counterparts. Still, the goal of reducing the random factor in planning while retaining the advantages of algorithms which exploit it is highly desirable. Algorithms that retain a high degree of performance and gain improved predictability are attractive for many applications.

Work on similar problems such as satisfiability and constraint satisfaction in other fields (Gomes et al. 2000; Gomes, Selman, and Kautz 1998) has exploited a feature common to various randomized algorithms: the tails of their runtime distributions can be such that the expected run cost of a new instance would be less than that required to complete the current instance. The presence of these so-called heavy-tailed distributions implies that even straightforward methods of controlling these algorithms at a high level, such as enforcing a restart threshold, can provide performance improvements. We propose to take advantage of the nature of the runtime distribution of the RRT in a similar way, with the goal of improving its performance in terms of both mean and variability of runtime.

We first provide an overview of research related to the use of sampling-based planning and the methodology for improving randomized algorithms in Background. We follow with a short justification of our expectation that RRT runtimes exhibit distributions that allow restarts to be of benefit in Motivation. The Theory of Restarts section introduces a mathematical characterization of restarts, which we utilize in Experiments to examine the runtime results of a variety of problems, ranging from a narrow tunnel to the 16-puzzle and the Alpha-puzzle. We also implement restarts in several problems that apply well over a variety of queries. Finally, we provide closing thoughts and future directions in Conclusions.

Background

Randomized, sampling-based planning algorithms were originally pioneered in an attempt to defeat the “curse of dimensionality.” One of the most successful of this class of algorithms is the RRT, which uses random sampling to

probabilistically bias the growth of a tree toward unexplored regions. It has been successfully applied to a range of problems of continuous, discrete, and hybrid type (Branicky et al. 2006; Morgan and Branicky 2004; Plaku, Kavraki, and Vardi 2007). However, understanding of the issues surrounding the use of randomization in planning is still an open question (Lindemann and LaValle 2004a).

Our work to improve RRT runtimes using appropriately-chosen restart thresholds are preceded by related works using other planning algorithms and problem types. Challou, Gini, and Kumar (1993) used multiple processors to run parallel copies of a randomized planning algorithm, taking the first result. This strategy lessens the impact of “unlucky” choices made in some instances of randomized algorithms. Isto, Mäntylä, and Tuominen (2003) developed heuristic-based variants of the PRM with a focus on reducing its run cost variance. Geraerts and Overmars (2004) achieved considerable improvements to variability in the PRM by using restarts in a constrained 3D wrench-moving problem. Presently, Carroll (2008) is investigating the use of restarts with a focus on the RRT and hybrid systems. We follow the spirit of these works by examining a restart strategy that, although designed in consideration of the RRT’s properties, treats it as a black-box.

Applying restarts to optimize the behavior of general Las Vegas algorithms (those with runtime as a random variable) is a mature concept, with existing methods for minimizing the expected run length of unknown problem instances (Luby, Sinclair, and Zuckerman 1993) and for minimizing tail probabilities (Alt et al. 1996). Gomes, Selman, and Kautz (1998) took advantage of these ideas to achieve dramatic improvements in the solving of Boolean satisfiability and constraint satisfaction problems. Streeter, Golovin, and Smith (2007b; 2007a) recently extended these concepts to sets of problem instances and heuristics.

Motivation

The RRT algorithm has a known exponential bound on its run length tail probabilities (LaValle and Kuffner 2001); though its use of a nearest neighbor operation implies that individual iterations require increasing time. Therefore, it is reasonable to assume that runtime tail probabilities may arise that are heavier than exponential. Figure 1 illustrates a potential instance of this phenomenon: two independent forward-search RRTs solving the same query (a source – the left dot and a destination – the right dot) can reach roughly the same level of progress at drastically different run lengths (166 versus 1887 iterations). In this case, the “unlucky” tree will take more time to solve the problem than the “lucky” one. Further, with continuing growth by random sampling, it is possible that this “unlucky” tree will require more time to complete than the entire instance of

the “lucky” one. The critical issue we examine is whether the RRT algorithm generates “lucky” and “unlucky” trees in such a way that will allow restarts to offer performance improvements, measured by runtime mean and variability.

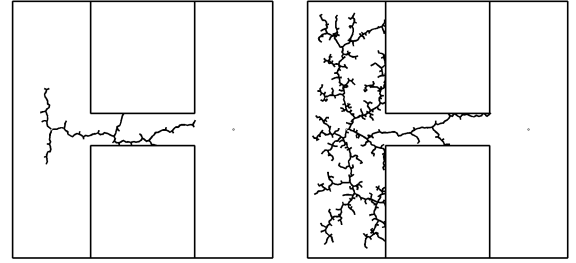


Figure 1: “Lucky” and “Unlucky” Instances of an RRT

Theory of Restarts

In a static environment with a fixed query, the runtime of a randomized algorithm is a stationary random variable, described with probability density $f_x(x)$ and cumulative distribution $F_x(x \leq x)$. Following the work of Luby, Sinclair, and Zuckerman (1993), suppose we define a *restart threshold*, β , such that the algorithm runs to the threshold and restarts from scratch if not completed. Furthermore, we define the probability that a run will fall below the threshold ($F_x(x \leq \beta)$) as $p_x(\beta)$. This restart strategy results in a new distribution, $f_r(x)$, that is composed of periodic, exponentially-scaled copies of the lower portion of the original distribution, as given in (1).

$$f_r(x, \beta) = (1 - p_x(\beta))^{\lfloor \frac{x}{\beta} \rfloor} f_x\left(x - \beta \lfloor \frac{x}{\beta} \rfloor\right) \quad (1)$$

We note that there are no terms corresponding to the tail of the original distribution, replacing its shape with that of (roughly) an exponential with base $1 - p_x(\beta)$ and decay rate β^{-1} . Similarly, this restart distribution has mean $\mu_r(\beta)$, given by (2), and variance $\sigma_r^2(\beta)$, given by (3), that are determined only by the first portion of the original distribution, its renormalized statistics (mean $\mu_-(\beta)$ and variance $\sigma_-^2(\beta)$), and the magnitude of the restart threshold. Under appropriate conditions, the mean and variance can be improved by this restart strategy.

$$\begin{aligned} \mu_-(\beta) &= \frac{1}{p_x(\beta)} \int_0^\beta x f_x(x) dx \\ \mu_r(\beta) &= \mu_-(\beta) + \left(\frac{\beta}{p_x(\beta)}\right) (1 - p_x(\beta)) \end{aligned} \quad (2)$$

$$\begin{aligned} \sigma_-^2(\beta) &= \frac{1}{p_x(\beta)} \int_0^\beta (x - \mu_-(\beta))^2 f_x(x) dx \\ \sigma_r^2(\beta) &= \sigma_-^2(\beta) + \left(\frac{\beta}{p_x(\beta)}\right)^2 (1 - p_x(\beta)) \end{aligned} \quad (3)$$

The range of restart thresholds that result in a reduction in the mean is given by (4). Setting the β -derivative of (2) to zero results in the optimal (viz. minimal) value of the restart distribution mean. This optimal restart mean, given in (5), is bounded above by the mean, since it is always possible to set the threshold to infinity and leave the distribution unchanged. However, it can be difficult to compute (5) accurately from sampled data due to the appearance of the $f_x(\beta^*)$ term. Instead, we compute the mean in (2) on a set of logarithmically-spaced thresholds in the sampled range and take the threshold with the smallest mean.

$$\mu_r(\beta) \leq \mu \Leftrightarrow \mu - \mu_-(\beta) \geq \beta \frac{1-F_x(x \leq \beta)}{F_x(x \leq \beta)} \quad (4)$$

$$\mu_r^*(\beta^*) = \frac{1-F_x(x \leq \beta^*)}{f_x(\beta^*)} \quad (5)$$

Finally, note that assuming equality in (4) and converting into a differential equation leads to (6), which shows that the family of exponential distributions remains unchanged by restarts. This is helpful as we can use the presence of tail probabilities with slower-than-exponential decay as a clue on when to apply restarts. A semilog plot of the runtime's survivor function ($1 - F_x(x \leq x)$), like Figures 5 and 7, can expose this tendency in a clear-cut way since the exponential survivor function appears linear.

$$\begin{aligned} \mu - \mu_-(\beta) &= \beta \frac{1-F_x(x \leq \beta)}{F_x(x \leq \beta)} \\ \Rightarrow f_x'^2(x) &= f_x''(x) f_x(x) \\ \Rightarrow f_x(x) &= \lambda e^{-\lambda x} \end{aligned} \quad (6)$$

Experiments

We conducted various simulations to obtain sampled cumulative distributions of wall clock runtime for RRT queries (both fixed and randomized) in five problem environments: Tunnel, Pincer, Passage, 16-puzzle, and Alpha-puzzle, which are described in the following subsections. In each setting, we analyze the potential of various restart strategies, observing as much as order of magnitude runtime mean and coefficient of variation improvements in specific cases. Figure 2 depicts the Tunnel, Pincer, Passage, and Alpha-puzzle problems along with a typical query for each.

The Pincer, Passage, and Alpha-puzzle were implemented and tested using the Object Oriented Programming System for Motion Planning (Plaku, Bekris, and Kavraki 2007; Plaku and Kavraki 2008), a recently-released software package that implements a variety of motion planning algorithms in a general framework. All tests use implementations of the bidirectional RRT that ran to

completion on an Intel Core™2 Duo E6850 system with 2GB of RAM. In general, we present the coefficient of variation (rather than variance) as a measure of runtime variability, since it is scale-invariant.

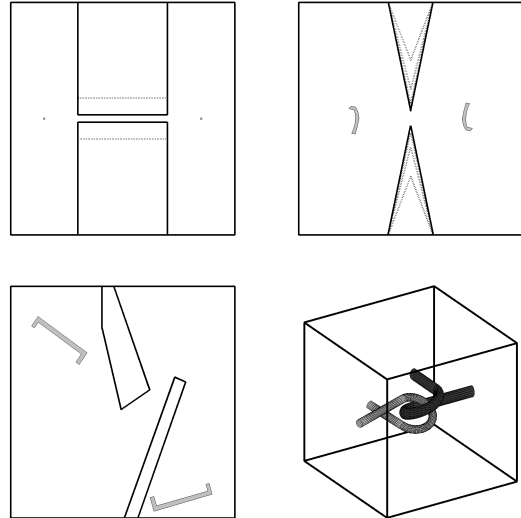


Figure 2: Test Problems with Typical Query States (Tunnel, Pincer, Passage, and Alpha-puzzle)

Tunnel and Pincer

The Tunnel problem environment is a straightforward 2D narrow-passage problem in which a point robot in $[0,1]^2$ must solve a fixed query that straddles a length 0.4 tunnel. An RRT solver with step size $\epsilon = 0.01$ was run 10,000 times for each of 16 tunnel widths ($2^{-n/2} : n \in \{0,1,\dots,15\}$). The results were used to compute an approximately-optimal restart threshold for each problem instance. This optimal restart threshold results in the minimum restart mean (seen as the minimum of a plot of (2), as in Figure 6). Each restart threshold fell between 0.5 and 5.0 ms, following an increasing trend with problem difficulty (narrowness of the tunnel). Figure 3 compares the baseline test results to the theoretical restart results from (2) and (3) and those from 10,000 restart simulations (per point). We also carried out simulations for 3D and 4D Tunnels, and the data was qualitatively parallel to that presented in Figure 3, though the divergence between the baseline and restart curves occurred at increasing tunnel widths.

Pincer is similar to Tunnel, with the exception that the agent is a 2D hook-shaped robot that operates in $SE(2)$. The experiments share all the same parameters, though the set of gap distances ($2^{-n/2} : n \in \{0,1,\dots,8\}$) is relaxed to account for the robot's nonzero width. Figure 4 presents the baseline data and the theoretical effects of implementing the optimal restart thresholds, which were of similar magnitude as in the Tunnel case: between 6.2 and 24.5 ms.

In both cases, we observe that implementing restarts has little to no benefit in the “easy” problems, in which the obstacle-free region in configuration space is large. However, as the gap shrinks, the potential of restarts becomes apparent as the performance measures separate. In the most restrictive Tunnel test, where the gap was approximately 0.5% as large as the space’s bounding box edge size, restarts provided a mean runtime advantage of over 20-fold while decreasing the coefficient of variation by 6-fold. The theoretical results for the Pincer tests display similar trends as the configuration space of the environment becomes more restricted.

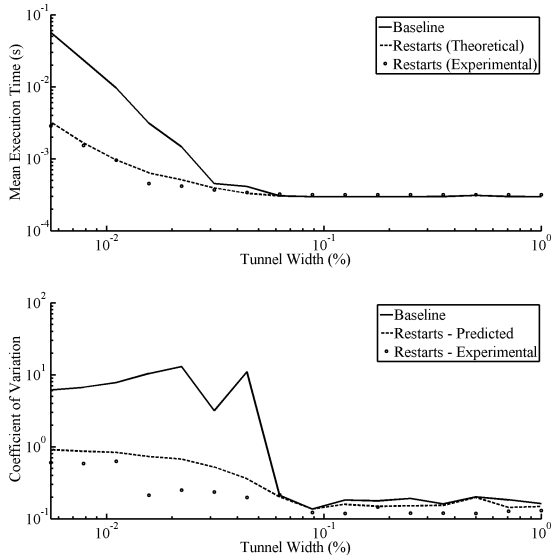


Figure 3: Runtime Means and Coefficients of Variation for Tunnel Tests

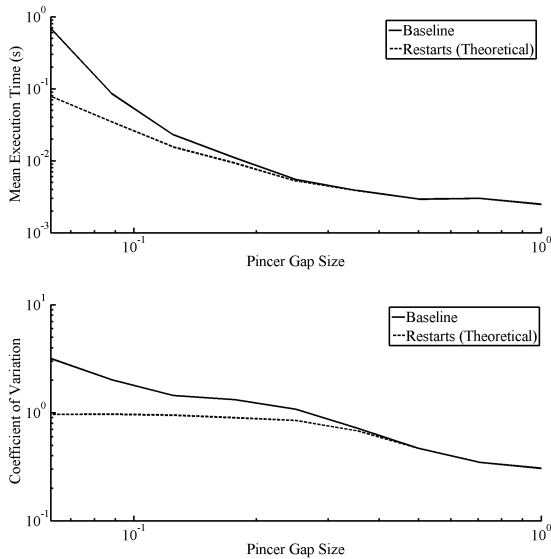


Figure 4: Runtime Means and Coefficients of Variation for Pincer Tests

Passage

Passage uses a 2D environment and an $SE(2)$ robot (StraightC) provided with OOPSMP to test whether a restart threshold for one problem instance will generalize to a variety of similar instances. First, we selected a single pairing in which both states were near the center of the open area and executed the solver for 1000 runs. From this data set, we computed an approximately-optimal restart threshold, which we then applied to solve a set of 1000 randomly-generated queries that required the robot to move through the passage from left to right. Table 1 lists the resulting statistics for these simulations (in units of seconds), including the 1st and 9th deciles, which measure the times at which 10% and 90% of runs have succeeded.

	Baseline	Restarts	Change
Mean	0.465	0.392	-15.8%
Variance	0.312	0.202	-35.3%
Coeff. of Var.	1.201	1.147	-4.5%
1 st Decile	0.057	0.051	-12.2%
Median	0.272	0.232	-14.9%
9 th Decile	1.089	0.956	-12.0%

Table 1: Experimental Statistics for Random Passage Queries

The potential performance benefit of restarts in this problem is moderate, due to the less restrictive passage in configuration space. The baseline data predicts a performance improvement of 23.1% for applying restarts for the tested pair of initial and final states. When applied to a diverse set of queries, restarts achieved a mean runtime speed increase of 18.7%, while the coefficient of variation was reduced by 4.5%. While the second set of simulations does not achieve the predicted level of improvement for the unique query, it does demonstrate that it is possible to benefit by applying a single restart threshold derived from a representative initial and final state to a set of queries in a given environment.

16-puzzle

In contrast to the previous problems, the 16-puzzle is a discrete search problem based on the common pictorial tile-shifting puzzle. With more than ten million million states ($16!/2 \approx 10^{13}$), this problem is intractable for many conventional discrete search algorithms such as breadth-first search. The discrete version of the RRT has been shown to outperform traditional A* in terms of memory and solution time (Morgan 2004; Morgan and Branicky 2004). Hence, we expect the 16-puzzle may be tractable for the discrete RRT using the traditional metric of the sum of each tile’s Manhattan distance between its current and goal positions. Figure 5 demonstrates this fact with a plot of the survivor function of the runtimes, showing that 99% of instances on a single, randomized initial state complete

within 100 seconds. This plot includes a set of exponential distribution survivor functions (straight lines on the semi-log axes); the RRT’s survivor function appears to have an exponential or worse tail, marking it as a possible candidate for restarts.

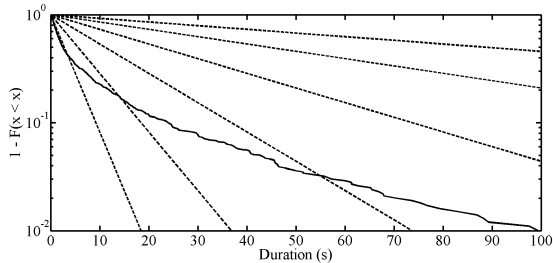


Figure 5: Runtime Distribution for a Single 16-puzzle Instance

As a precursor to applying restarts to the general 16-puzzle problem, we first examine their potential in this single initial state case. Figure 6 compares the baseline runtime mean and coefficient of variation to the theoretical restart values on a 1,000-sample data set. For the optimal restart threshold, this implies a 2.9-fold mean runtime improvement and a halving of the coefficient of variation.

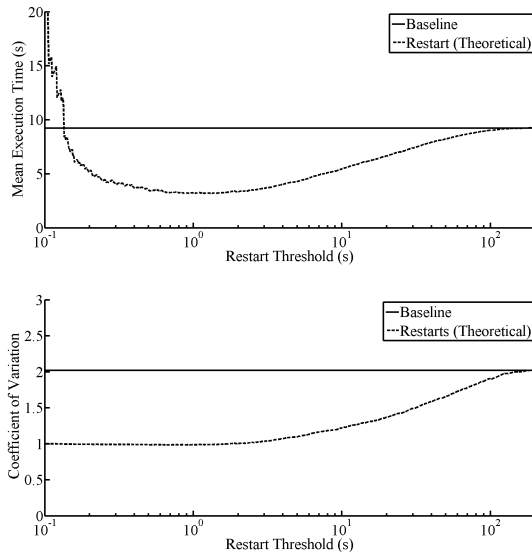


Figure 6: Runtime Means and Coefficients of Variation for a Single 16-puzzle Instance

In addition to improving solution runtime and variability for one specific instance of the 16-puzzle, we would like to be able to provide a solution that allows an average-case improvement for a variety of initial states. We collected 10,000 samples of the RRT solutions to 16-puzzle, using a randomized (but solvable) initial state in each case. This generates an aggregated distribution of solution runtimes across the 16-puzzle. Note that we cannot necessarily use

this distribution to compute an optimal restart threshold that will apply well across all 16-puzzle initial states. This approach implicitly assumes that restarting on a given query could generate a runtime from any query, which is troubling if the two queries have solutions of drastically different lengths. In this case, their individual distributions should not be well-approximated by the aggregate distribution. Still, if the aggregate distribution does provide a reasonable approximation to the set of distributions for each query, we expect the resulting restart threshold will apply well.

We tested three different methods of computing a restart threshold for the 16-puzzle. First, we simply take the optimal value for the aggregate distribution (466 ms). Second, we separate the distribution by each (discrete) value of the metric distance between initial and final states and compute a restart threshold for each one (on average, 881 ms). Note that in this case, the small number of samples for some metric values creates false minimums on the restart threshold means plot and results in prohibitively small values. In order to avoid endlessly restarting certain cases, we also enforce a minimum restart threshold of the aggregate optimal value. Finally, we use the set of metric-based values to compute a mean restart threshold (415 ms) that is weighted in proportion to the number of samples for each metric value. The restart threshold from the aggregate distribution predicts a 3.7-fold improvement in mean runtime, which can be taken as an upper bound on the expected performance of these strategies.

	Baseline	Aggregate	Weighted	Metric
Mean	13.54	5.01	6.37	5.85
Variance	810.7	59.8	133.9	63.4
Coeff. of Var.	2.10	1.54	1.82	1.40
1 st Decile	0.39	0.28	0.41	0.35
Median	3.81	2.67	3.34	3.14
9 th Decile	35.7	12.0	14.2	14.0

Table 2: Experimental Statistics for Random 16-puzzles

Table 2 presents the results of running sets of 1,000 initial states with each restart strategy (in units of seconds). All three provide notable improvements in both mean and coefficient of variation. Somewhat surprisingly, the single aggregate restart threshold provides the best mean runtime improvement of 2.7-fold, though the individual metric-based restart thresholds perform slightly better in terms of coefficient of variation. All three strategies provide major improvements over the baseline data by most measures. The one exception is the 1st decile, which we expect to be unaffected by restart thresholds that fall above it. In contrast, the 9th decile is greatly improved, since restarts make extremely long runtimes much less probable without compromising the potential for extremely short ones.

Alpha-puzzle

The final test environment is the Alpha-puzzle, a common disassembly problem with a narrow passage used to benchmark motion planning algorithms. The 3D models used are publicly available (Yamrom 1999). This problem has the most significant of the runtimes tested, as its agent has six degrees of freedom and the most complex collision-checking. Due to our need to collect a representative sample of solutions runtimes, we use a version of the traditional puzzle that is scaled by a factor of 1.5 in order to widen the passage and reduce solution times. In this puzzle, the vertical gap between the crossed portions of the obstacle is approximately 18% greater than the agent’s thickness. This allows a solution in which a portion of the agent is simply slid through the gap. This type of solution is also possible in the 1.2-scaled version, though this measure is reduced to approximately 5%, while the standard version requires a plan that aligns the crossed portions of the agent and obstacle in a more complex way.

The distribution of 2000 runtimes for the Alpha-puzzle is shown in Figure 7 and is very close to the accompanying exponential fit. Accordingly, we expect restarts to yield a minimal potential return on this query. Here, the optimal restart threshold (153.3 s) leads to an improvement in mean runtime of 2.3% and in coefficient of variation of 4.8%. Though we also attempted the 1.2-scaled Alpha-puzzle, solutions repeatedly failed to complete before consuming all available memory of the test machine (and nearly 45 minutes). However, based on the results of the Tunnel and Pincer experiments, we expect that less scaled versions would provide greater potential improvement.

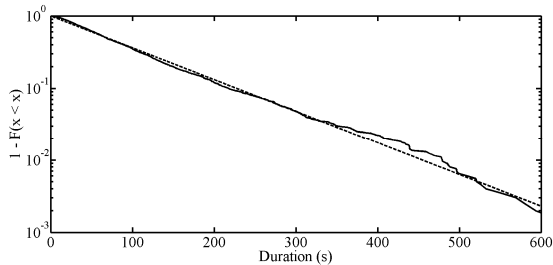


Figure 7: Runtime Distribution for the 1.5-scaled Alpha-puzzle

We note that a parallel strategy like that of Challou, Gini, and Kumar (1993), in which a set of planners terminates with the first solution, also fails on this Alpha-puzzle data. Few planners result in small increases in mean runtime (4.3% for two planners and 8.9% for three), with further degradation for larger numbers. Table 3 presents the theoretical statistics for the use of this strategy on the Alpha-puzzle data (in units of seconds). In this case, there are fair benefits in variability for few planners, though this comes at the cost of increased mean runtime.

	Baseline	Two	Three	Four
Mean	98.7	102.9	107.5	112.9
Variance	9652.4	8416.0	8195.5	8512.1
Coeff. of Var.	0.995	0.891	0.842	0.817
1 st Decile	15.6	20.4	20.6	23.2
Median	67.3	77.5	83.2	89.6
9 th Decile	217.2	216.9	220.3	232.2

Table 3: Theoretical Statistics for Alpha-puzzle Parallel Planners

Conclusions

We have examined the performance of the RRT algorithm on a variety of problem environments and provided results on the degree of benefit provided by applying restarts. Our experiments demonstrate a definite tendency for restarts to become more useful as the “difficulty” of a particular planning problem increases. Additionally, we have shown that cases such as the Passage and the 16-puzzle are amenable to a single, generalized restart threshold that will provide runtime speed increases across varied queries. Though not all problems are well-suited to the restart strategy, these results provide motivation to further understand the complex issues introduced by randomized planning algorithms.

Carroll (2008) observes heavy-tailed distributions in other examples, including motion planning for a hovercraft and stabilization of an inverted pendulum. In the first case, the use of a set of solvers with differing step sizes leads to improvements in the overall probability of success with fixed total number of nodes. Clearly, there are many potential methods of leveraging the features of the RRT’s run cost distribution. Furthermore, there are a variety of applicable performance measures: runtime, total iterations, total nodes, probability of success, memory usage, and more. A significant open question then relates to how to optimize these measures in an adaptive and robust way, without a priori knowledge of the problem.

In particular, we believe that more attention is merited in the discrete case, in which the application of restarts provided a significant and pervasive benefit over varied queries in the 16-puzzle. While it remains to be seen if similar results would be observed on other discrete problems of similar and greater complexity, our experiments show promise for the concept. More advanced strategies that avoid discarding past progress as the restart strategy does could lead to even more significant changes in runtime and variability for these and other problems.

Acknowledgements

This work was supported by the Air Force Office of Scientific Research via an NDSEG Fellowship. We thank the authors of the OOPS for Motion Planning software.

References

- Alt, H.; Guibas, L.; Mehlhorn, K.; Karp, R.; and Wigderson, A., 1996. A Method of Obtaining Randomized Algorithms with Small Tail Probabilities. *Algorithmica* 16:543-547.
- Branicky, M. S.; Curtiss, M. M.; Levine, J.; and Morgan, S. 2006. Sampling-based Planning, Control, and Verification of Hybrid Systems. *IEEE Proceedings Control Theory and Applications* 153:575-590.
- Branicky, M. S.; LaValle, S. M.; Olson, K.; and Yang, L. 2001. Quasi-randomized Path Planning. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 1481-1487. Los Alamitos, CA, USA: IEEE Publications.
- Carroll, S. 2008. Verification Techniques for Hybrid Systems Using Rapidly-exploring Random Trees. M. S. thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH, USA. Forthcoming.
- Challou, D. J.; Gini, M.; and Kumar, V. 1993. Parallel Search Algorithms for Robot Motion Planning. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, 46-51. Los Alamitos, CA, USA: IEEE Publications.
- Geraerts, R.; and Overmars, M. H. 2004. Sampling Techniques for Probabilistic Roadmap Planners. In *Proceedings of the 8th International Conference on Intelligent Autonomous Systems*, 600-609. Amsterdam, Netherlands: IAS Society.
- Gomes, C. P.; Selman, B.; Crato, N.; and Kautz, H. 2000. Heavy-tailed Phenomena in Satisfiability and Constraint Satisfaction Problems. *Journal of Automated Reasoning* 24: 67-100.
- Gomes, C. P.; Selman, B.; and Kautz, H. 1998. Boosting Combinatorial Search through Randomization. In *Proceedings of the 15th Conference on Artificial Intelligence*, 431-437. Menlo Park, CA, USA: AAAI Press.
- Isto, P.; Mäntylä, M.; and Tuominen, J. 2003. On Addressing the Run-cost Variance in Randomized Motion Planners. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2934-2939. Los Alamitos, CA, USA: IEEE Publications.
- Kavraki, L. E.; Švestka, P.; Latombe, J.; and Overmars, M. H. 1996. Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation* 12:566-580.
- LaValle, S. M. 1998. Rapidly-exploring Random Trees: A New Tool for Path Planning, Technical Report, TR 98-11, Computer Science Department, Iowa State University.
- LaValle, S. M.; Branicky, M. S.; and Lindemann, S. R. 2004. On the Relationship Between Classical Grid Search and Probabilistic Roadmaps. *International Journal of Robotics Research* 23:673-692.
- LaValle, S. M.; and Kuffner, J. J. 2001. Rapidly-exploring Random Trees: Progress and Prospects. In *Algorithmic and Computational Robotics: New Directions*, 293-308. Boston, MA, USA: A. K. Peters.
- Lindemann, S. R.; and LaValle, S. M. 2004a. Current Issues in Sampling-based Motion Planning. In *Proceedings of the 8th International Symposium on Robotics Research*, 36-54. Berlin, Germany: Springer-Verlag.
- Lindemann, S. R.; and LaValle, S. M. 2004b. Steps toward Derandomizing RRTs. In *Proceedings of the 4th International Workshop on Robot Motion and Control*, 271-277. Los Alamitos, CA, USA: IEEE Publications.
- Luby, M.; Sinclair, A.; and Zuckerman, D. 1993. Optimal Speedup of Las Vegas Algorithms. In *Proceedings of the 2nd Israel Symposium on the Theory of Computing and Systems*, 128-133. Los Alamitos, CA, USA: IEEE Publications.
- Morgan, S. 2004. Sampling-based Planning for Discrete Spaces. M. S. thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH, USA.
- Morgan, S.; and Branicky, M. S. 2004. Sampling-based Planning for Discrete Spaces. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1938-1945. Los Alamitos, CA, USA: IEEE Publications.
- Plaku, E.; Bekris, K. E.; and Kavraki, L. E. 2007. OOPS for Motion Planning: An Online, Open-source Programming System. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, 3711-3716. Los Alamitos, CA, USA: IEEE Publications.
- Plaku, E.; and Kavraki, L. E. 2008. "OOPSMP: Object-oriented Programming System for Motion Planning," *Rice University Kavraki Lab*, February, 2008 [Online]. Available: <http://www.kavrakilab.org/OOPSMP/index.html>. [Accessed March 10, 2008].
- Plaku, E.; Kavraki, L. E.; and Vardi, M. Y. 2007. A Motion Planner for a Hybrid Robotic System with Kinodynamic Constraints. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, 692-697. Los Alamitos, CA, USA: IEEE Publications.
- Streeter, M.; Golovin, D.; and Smith, S. F. 2007a. Combining Multiple Heuristics Online. In *Proceedings of the 22nd Conference on Artificial Intelligence*, 1197-1203. Menlo Park, CA, USA: AAAI Press.
- Streeter, M.; Golovin, D.; and Smith, S. F. 2007b. Restart Schedules for Ensembles of Problem Instances. In *Proceedings of the 22nd Conference on Artificial Intelligence*, 1204-1210. Menlo Park, CA, USA: AAAI Press.
- Yamrom, B. 1999. "Alpha Puzzle," *Texas A&M University Algorithms & Applications*, June, 1999 [Online]. Available: <http://parasol-www.cs.tamu.edu/dsmft/benchmarks/>. [Accessed March 24, 2008].