Risk-Aware Graph Search with Dynamic Edge Cost Discovery

Ryan Skeele, Jen Jen Chung, Geoffrey A. Hollinger

Oregon State University, Corvallis, OR, USA
{skeeler,jenjen.chung,geoff.hollinger}@oregonstate.edu

Abstract. In this paper we introduce a novel algorithm for incorporating uncertainty into lookahead planning. Our algorithm searches through connected graphs with uncertain edge costs represented by known probability distributions. As a robot moves through the graph, the true edge costs of adjacent edges are revealed to the planner prior to traversal. This locally revealed information allows the planner to improve performance by predicting the benefit of edge costs revealed in the future and updating the plan accordingly in an online manner. Our proposed algorithm, Risk-Aware Graph Search (RAGS), selects paths with high probability of yielding low costs based on the probability distributions of individual edge traversal costs. We analyze RAGS for its correctness and computational complexity and provide a bounding strategy to reduce its complexity. We then present results in an example search domain and report improved performance compared to traditional heuristic search techniques. Lastly, we implement the algorithm on satellite imagery to show the benefits of risk-aware planning through uncertain terrain.

Keywords: Risk-Aware Planning, Graph Search, Planning Under Uncertainty

1 Introduction

When planning in unstructured environments there is a greater need for fast, reliable path planning methods capable of operating under uncertainty. Planning under uncertainty allows for robustness when faced with unknown and partially known environments. We introduce a method, Risk-Aware Graph Search (RAGS), for finding paths through graphs with uncertain edge costs (similar to Stochastic Shortest Path Problems with Recourse [1]). In the domain of interest, a robot moves through an environment represented by a graph, and the true costs for edges adjacent to the robot's location are dynamically revealed. Our method accounts for both the uncertainty in the edge costs and the dynamic revealing of these costs. Thus, we bridge the gap between traditional search methods [2,3] and risk-aware planning under uncertainty [4,5].

Traditionally, graphs are composed of nodes and edges, with a node representing some state and an edge representing the transition between states. Effectively searching through a graph with known edges has been extensively researched and applies across disciplines in robotics, computer science, and optimization. We aim to expand the capabilities of graph search algorithms by allowing for uncertainty in traversal costs and dynamic discovery of those costs,

while avoiding the blowup in computation from more expressive frameworks (e.g., POMDPs). Our novel approach searches over uncertain edge costs with known distributions and properly adjusts as new information about the edge costs becomes available. This formulation allows for computationally efficient methods of reducing the risk of traversing paths with high cost.

The main novelty of this work is the introduction of a non-myopic graph search algorithm for risk-aware planning with uncertain edge costs and dynamic local edge discovery. RAGS is an online planning mechanism that incorporates live feedback for deciding when to be conservative and when to be aggressive. With edge costs modeled as probability distributions, we can derive a principled way of leveraging information further down a path. This leads to a tradeoff between revealing the true cost of a large number of edges (exploration) and traversing uncertain edges with low mean cost (exploitation). RAGS addresses this tradeoff in a principled way, and the result is a low probability of executing a path with high cost.

We compare our method to *A**, *sampled A** [5], and a greedy approach on a large number of random connected graphs. The results show that RAGS reduces the number of high cost runs compared to all other tested methods. In addition to testing over random graphs, we validate RAGS using satellite imagery. By applying a series of filters to satellite data, we are able to extract potential obstacles that may impede a robot moving through the map. To deal with the imprecise nature of obstacle extraction, we can utilize the benefits of RAGS to find paths that are less likely to be substantially delayed. We show examples of different cases in which RAGS finds preferable paths. The algorithm is run over 100 satellite images taken from a broad set of landscapes. The resulting path costs over the image database confirms the benefit of the RAGS algorithm in a real-world planning domain.

The remainder of this paper is organized as follows. First we review related work and research in probabilistic planners (Section 2). We then introduce the path search problem with uncertain traversal costs and dynamic edge cost discovery (Section 3). Next we derive a method for quantifying path risk (Section 4.1) and present a way to reduce the search space by removing dominated paths (Section 4.2). In Section 4.3 we present the RAGS algorithm and show comparisons to existing search algorithms in Section 5. We then highlight the utility of RAGS by demonstrating its effectiveness for planning through various terrain captured from satellite imagery (Section 6). Finally, we draw conclusions and propose future directions for this line of research (Section 7).

2 Related Work

Planning under uncertainty is a challenging problem in robotics. An underlying assumption in many existing planning algorithms is that the search space is not stochastic. Under this assumption, researchers have developed many powerful algorithms like *A** [2] and *RRT** [6] that perform efficient point to point planning over expected costs (see [7] and [8] for surveys). These algorithms are

efficient for problems with deterministic actions and a well-defined search space, but are often not well suited to planning under uncertainty. Recent work has explored ways of incorporating uncertainty into similar planners in field robotics applications [4,9].

Reasoning probabilistically in robotic planning allows performance to degrade gracefully when encountering the unexpected. Notable work has been done on incorporating uncertainty from sensors into the state estimation of the system, [10,11], or in the path planning itself [12,13]. However, uncertainty can lie in both the state and the world model, so we must address both sources of uncertainty to plan effectively.

Prior work in uncertain traversability of graph edges has focused on a binary status of the edge. This family of problems is a variant of the shortest path problem known as the Canadian Traveler Problem (CTP) [14]. The CTP is inspired by drivers in northern Canada who sometimes have to deal with snow blocking roads and causing delays. The focus of CTP, and variations like it is to plan paths/policies when graph connections are uncertain. A slight variation, known as Stochastic Shortest Path with Recourse (SSPR) [1], adds random arc costs. Our problem formulation, similar to the CTP and SSPR, provides local information during traversal. This is also consistent with the algorithm *PAO** for domains where there is a hidden state in the graph [15]. The hidden state relates this work to Partially Observable Markov Decision Process (POMDPs) [16], which provides an expressive framework for uncertain states, actions, and observations. In our case, we assume there is only uncertainty in the transition costs, which avoids computational blowup often found in large POMDPs.

Planning over the expected cumulative cost is another relevant variant of the shortest path problem [17]. Risk-Sensitive Markov Decision Process are one approach to such stochastic planning problems. These solutions address cases where large deviations from the expected behavior can have detrimental effects [18]. Previous approaches have used a parameter for risk aversion to solve Risk-Sensitive MDPs [19]. Like these techniques, we aim to avoid large deviations from the expected value while planning for low costs; however we also look to exploit local information available during execution. We build on work in risk-aware planning (e.g., Risk-Sensitive MDPs), which deal with probability distributions over outcomes. Other risk-aware planning techniques in the literature use bounding of likelihood [20] by minimizing the path length with respect to a lower bound on the probability of success. This is similar to work in [21], which instead bounds the average risk. These algorithms define reasonable ways of assigning a value for trading off between risk and a primary search objective like distance, but they do not incorporate dynamically revealed information as part of the search.

The stochastic edge cost problem has been approached using an iterative sampling method when dealing with uncertainty in terrain classification [5]. In this prior work, the edge values between landmarks are sampled from modeled cost distributions, and $A*$ is used over each sampling to generate a list of paths. The paths most frequently taken are considered the most likely to return low cost

paths, which yields a method (*sampled A**) that we test our algorithm against. In our work, we derive a formula for reducing the risk of a path based on the uncertainty of the traversal costs themselves, which allows for a more expressive framework than heuristic searches and reduced computational requirements than sampled approaches.

3 Problem Formulation

In this paper, we consider the problem of planning and executing a risk-aware path through an uncertain environment where knowledge of the true traversal costs are revealed only as we arrive within some proximity of an area. This planning scenario can be described over a graph with edge costs initially represented by some set of distributions, with the true edge costs identified as we arrive at the parent node of each edge during the path execution. Although we focus on a path cost minimization objective in this paper, we note that this formulation could represent informative path planning objectives by considering the equivalent maximization problem [22]. We now formulate the path search problem with uncertain edge costs and introduce notation that will be used throughout the paper.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the cost of traversing edge $E \in \mathcal{E}$ is drawn from a normal distribution $\mathcal{N}\left(\mu_E, \sigma_E^2\right)$. The cost of a path $\mathcal{P} \subset \mathcal{E}$ is the sum of the edge traversal costs, each of which are normally distributed. Thus the total path cost is drawn from $\mathcal{N}\left(\mu_\mathcal{P}, \sigma_\mathcal{P}^2\right)$, where $\mu_\mathcal{P} = \sum_\mathcal{P} \mu_E$ and $\sigma_\mathcal{P}^2 = \sum_\mathcal{P} \sigma_E^2$. This formulation is similar to that of the random network used in [1].

Assumption 1 *The true edge costs are drawn as i.i.d. samples from the respective cost distributions when queried.*

An important implication of Assumption 1 is that although multiple paths may share a subset of edges, all paths can be treated as having independent cost distributions. This is analogous to a problem where traversal costs are not fixed, but instead vary over time, and edge costs are queried upon arrival at the parent node. This assumption simplifies the computation of risk and dominance (shown later in Section 4) without significantly changing the nature of the problem.

Given any pair of start and goal vertices in the graph, $V_s, V_g \in \mathcal{V}$, the task is to traverse the graph along the *acyclic* path of least risk. More precisely, since each edge transition prunes the available set of paths to the goal, the path of least risk retains the highest probability of traversing the overall least-cost path as each transition is executed from V_s to V_g.

The decision at each vertex can be formulated by considering the next available transitions and their associated path sets. For example, say edge connections exist between the current vertex V_t and each of the vertices in the set $\mathcal{V}_{t+1} = \{A, B, C, \cdots\}$; furthermore, m acyclic paths exist from vertex A to V_g, while n acyclic paths exist from vertex B to V_g, etc. Let $\mathcal{A}$ be the set of random variables A_i, $i = \{1, \cdots, m\}$, where $A_i \sim \mathcal{N}\left(\mu_{A_i}, \sigma_{A_i}^2\right)$ represents the cost distribution of path i from vertex A to V_g (see Figure 1). Let c_{A_i} be a sample of

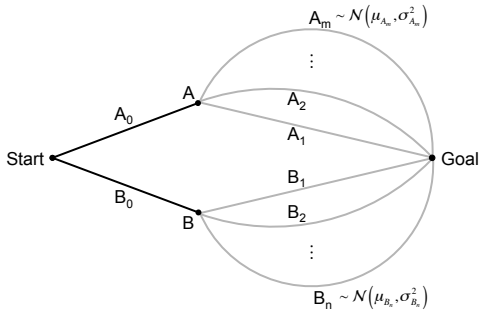


Fig. 1. Setup of pairwise comparison for sequential lookahead planning. Given the path cost distributions, we can directly compute the probability that traveling from Start to Goal through A will ultimately yield a cheaper path than traveling via B.

the random variable A_i and let c_{A_0} be the known cost of transitioning between V_t and A, then the lowest-cost path from V_t to A and onwards to V_g has cost:

$$c_{A_{min}} = c_{A_0} + \min_{A_i \in \mathcal{A}} c_{A_i}. \tag{1}$$

Similar statements can be made for path sets $\mathcal{B}, \mathcal{C}, \cdots$. To traverse the path of least-risk, each edge transition must select the next vertex $V \in \mathcal{V}_{t+1}$ such that the following probability holds,

$$P\left(c_{V_{min}} < c_{V'_{min}}\right) \geq 0.5, \quad \forall V' \in \mathcal{V}_{t+1} \setminus V. \tag{2}$$

That is, transitioning to vertex V results in a higher probability of executing a lower cost path than transitioning to any other neighboring vertex V'.

The pairwise comparisons between the available path sets obey a total ordering of preference. That is, given

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = \gamma,$$
$$P\left(c_{A_{min}} < c_{C_{min}}\right) = \epsilon,$$

then,

$$\gamma \leq \epsilon \Leftrightarrow P\left(c_{B_{min}} < c_{C_{min}}\right) \geq 0.5,$$

with equality iff $\gamma = \epsilon$. Thus $|\mathcal{V}_{t+1}| - 1$ pairwise comparisons are needed to solve for the next vertex V using (2).

4 Risk-Aware Graph Search (RAGS)

4.1 Quantifying Path Risk

The pairwise comparison $P\left(c_{A_{min}} < c_{B_{min}}\right)$ describes the probability that the lowest-cost path in the set $\mathcal{A}$ is cheaper than the lowest-cost path in the set $\mathcal{B}$.

Given Assumption 1, this probability can be expanded to,

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = \int_{-\infty}^{\infty} P\left(c_{B_{min}} = x\right) \cdot P\left(c_{A_{min}} < x\right) \mathrm{d}x. \tag{3}$$

We can now express each term in the integral according to the path cost distributions of each respective set. Let,

$$f(x, \mathcal{A}) = P\left(c_{A_{min}} < x\right),$$
$$g(x, \mathcal{B}) = P\left(c_{B_{min}} = x\right).$$

Since each of the path costs are drawn from normal distributions, then

$$f(x, \mathcal{A}) = 1 - \prod_{i=1}^{m} \tfrac{1}{2}\mathrm{erfc}\left(d(A_i)\right), \tag{4}$$

$$g(x, \mathcal{B}) = \sum_{j=1}^{n} \left[\tfrac{1}{\sqrt{2\pi}\sigma_{B_j}} \exp\left(-d(B_j)^2\right) \cdot \prod_{\substack{k=1 \\ k \neq j}}^{n} \tfrac{1}{2}\mathrm{erfc}\left(d(B_k)\right) \right], \tag{5}$$

where $d\left(\zeta_i\right) = \frac{x - c_{\zeta_0} - \mu_{\zeta_i}}{\sqrt{2}\sigma_{\zeta_i}}$.

As an aside, note that $f'(x, \cdot) = g(x, \cdot)$. Thus, using integration by parts, we can show that

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = 1 - P\left(c_{B_{min}} < c_{A_{min}}\right),$$

confirming that these two events are indeed complementary.

Equations (4) and (5) give some intuitive insight into the calculation of path risk. This formulation performs a tradeoff between the number of available paths in each set and the quality of the paths in each set, the latter represented by the means and variances of the path cost distributions. For example, $f(x, \mathcal{A})$ calculates the probability that the best path in $\mathcal{A}$ has a path cost less than x (the plot of $f(x, \mathcal{A})$ is shown in Figure 2 for $\mu_{\mathcal{A}_0} = \{20\}$ and $\sigma_{\mathcal{A}_0} = \{5\}$ as well as six other path set variants). From (4), we note that as $m \to \infty$, $f(x, \mathcal{A}) \to 1$, $\forall x \in (-\infty, \infty)$ and this is shown in the plots of $f(x, \mathcal{A}_{1,2})$. Path set $\mathcal{A}_1$ has twice as many paths (of equal means and variances) as $\mathcal{A}_0$, while $\mathcal{A}_2$ has three times as many. The curves show a trend towards an earlier and more rapid transition to 1 as m increases; however it is also apparent that adding more paths results in diminishing returns in terms of driving $f(x, \mathcal{A}) \to 1$, $\forall x \in (-\infty, \infty)$.

Other trends can be observed when adding paths of varying cost distributions to the set. $\mathcal{A}_3$ includes a second path with lower mean and equivalent variance to $\mathcal{A}_0$. The corresponding $f(x, \mathcal{A}_3)$ is shifted significantly to the left of $f(x, \mathcal{A}_0)$, causing the transition towards 1 to occur much earlier. On the other hand, the addition of a second path with higher mean and equivalent variance results in almost no change to the curve, as shown by the overlap between $f(x, \mathcal{A}_4)$ and

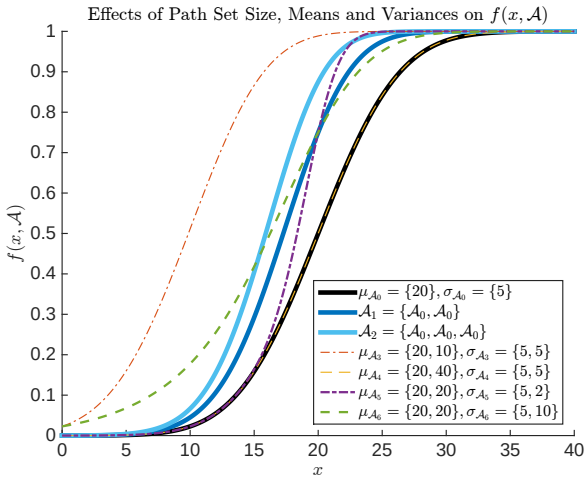


Fig. 2. The plot of $f(x, \mathcal{A})$ for seven path sets, $\mathcal{A}_0$ has a single path with cost drawn from $\mathcal{N}\left(20, 5^2\right)$. This plot shows the effects of including additional paths to the set whose costs are drawn from varying distributions.

$f(x, \mathcal{A}_0)$. Adding a path with equivalent mean but a lower variance results in the plot of $f(x, \mathcal{A}_5)$, which shows a much more rapid transition to 1 compared to either $f(x, \mathcal{A}_0)$ or $f(x, \mathcal{A}_1)$. However, the addition of a path with equal mean and higher variance increases the overall uncertainty associated with the path set, as shown by the shallower gradation in the probability curve. Furthermore, this means that although the transition towards 1 begins at lower values of x, as $x \to \infty$, $f(x, \mathcal{A}_6) \to 1$ more slowly than for $f(x, \mathcal{A}_1)$ or $f(x, \mathcal{A}_5)$.

This analysis motivates a bounding approach that compares the values of path cost means and variances to retain only the most relevant paths for the calculation of (3). Bounding the path set is especially desirable since the computation of each pairwise comparison has complexity $\mathcal{O}\left(n^2 m\right)$, where n and m are the sizes of the path sets under consideration.

4.2 Non-Dominated Path Set

Existing graph search algorithms, such as A*, use a total ordering of vertices based on the calculated cost-to-arrive to find a single optimal path between defined start and goal vertices. In contrast, an implementation of RAGS requires two sweeps of the graph, since an initial pass is required to gather cost-to-go information at each node for quantifying the path risk at execution. If all possible acyclic paths between start and goal are to be considered, then this initial pass is exponential in the average branching factor of the search tree. To reduce this computation, we introduce a partial ordering condition based on the path cost means and variances to sort the priority queue and terminate the search. This bounds the set of resultant paths to be considered during execution by accepting only those that exhibit desirable path cost mean and variance characteristics.

As discussed in Section 4.1, the addition of paths with higher mean costs to the existing path set results in little improvement in the overall risk of committing to that set. Similarly, adding paths with higher variances on the path cost results in a slower convergence of $f(x, \mathcal{A})$ to 1 as well as greater uncertainty regarding the true path cost outcomes for the set. Thus, an intuitive bounding condition is to only accept paths with lower path cost means and/or lower path cost variances. Furthermore, the same condition can be applied to the cost-to-arrive distributions to provide a partial ordering for the node expansions.

In practice, the partial ordering considers whether a path is *dominated* by an existing path, either in the open or closed set. For any two paths, A and B,

$$A \succ B \Leftrightarrow (\mu_A < \mu_B) \wedge \left(\sigma_A^2 < \sigma_B^2\right). \tag{6}$$

That is, if both the mean and variance of the cost of path B are greater than those of path A, then path A is said to *dominate* path B. This is a similar method to the one described in [13]. Using (6) to sort the priority queue guarantees that all non-dominated nodes are expanded before any dominated nodes are considered and only non-dominated paths to the goal vertex are accepted. This also allows the search to terminate once all paths in the open set are dominated by paths in the closed set. Thus the set of accepted paths from the initial search through the graph is referred to as the *non-dominated path set*. It is worth noting that the non-dominated path set is not guaranteed to contain the true optimal path; however it is guaranteed to contain the A* path calculated on the mean.

Proposition 1. *The non-dominated path set includes the A* path calculated on the mean of the edge cost distributions.*

Proof. The A* path calculated on the mean is the path with the lowest total mean cost. This path cannot be dominated by another path since the first inequality on the right hand side of (6) will never be true. □

4.3 RAGS Dynamic Execution

Given the non-dominated path set, path execution can occur by conducting edge transitions at each node to select the path set of least risk according to (2). The true edge transition costs, which become available for all neighboring edges to the current node, are included by directly substituting the known value of c_{V_0} into (1).[1] The pseudo code for the complete RAGS algorithm is provided in Algorithm 1; an initial sweep of the graph is conducted to search for the non-dominated path set while a second sweep is conducted during execution to incorporate edge cost information as it is received.

[1] Note that knowledge of the true neighboring edge costs is not required for RAGS to formulate a path. The immediate transition costs c_{A_0} and c_{B_0} can be included as distributions in (3) to determine a path from start to goal. Dynamic replanning is only necessary if new edge cost information is discovered.

Algorithm 1 Risk-Aware Graph Search

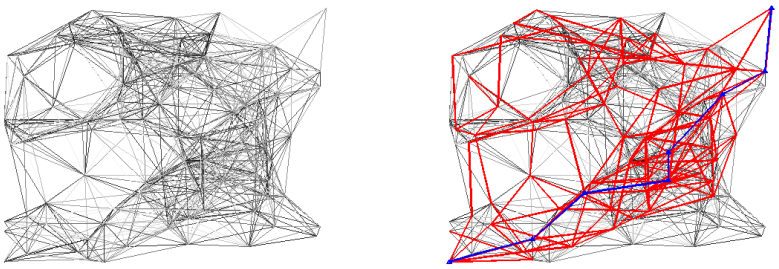
```
    // Initial sweep
 1: Initialize open, closed, N ← ∅        ▷ Initialize open and closed sets, and node data
 2: V_s ← start, V_g ← goal
 3: N.append(V_s)
 4: open.push(N)                                  ▷ Place the start node in the open set
 5: while open ≠ ∅ do
 6:     N_0 ← open.pop()                                          ▷ Current search node
 7:     V_{t+1} ← getNeighbors(N_0, G)
 8:     for V in V_{t+1} do                              ▷ Assess all neighboring vertices
 9:         N ← N_0.append(V)                                     ▷ Compute child node
10:         if notAncestor(V, N_0) ∧ nonDom(N, closed) then
11:             if V = V_g then
12:                 closed.push(N)
13:             else
14:                 open.push(N)                ▷ Open set sorted according to dominance
15:     if ¬nonDom(open.top(), closed) then     ▷ End search if all nodes in the open
16:         break                                 set are dominated by the closed set

    // Path execution
17: G_ND ← closed                  ▷ Directed graph formed by non-dominated path set
18: N ← ∅
19: V_0 ← V_s
20: while V_0 ≠ V_g do
21:     N.append(V_0)
22:     V_{t+1} ← getNeighbors(N, G_ND)
23:     V_ordered ← ComparePathSets(V_{t+1}, G_ND)   ▷ Total ordering of vertices from (3)
24:     V_0 = V_ordered.pop()                                  ▷ Execute edge traversal
```

5 Comparison to Existing Search Algorithms

We compared RAGS against a *naïve A** implementation, a *sampled A** method, and a *greedy* approach. *Naïve A** finds and executes the lowest-cost path based on the mean edge costs and does not perform any replanning. The *greedy* search is performed over the set of non-dominated paths and selects the cheapest edge to traverse at each step. The *sampled A** method searches over graphs constructed by sampling over the edge cost distributions and executes the path which is most frequently found. To provide a fair comparison, the planning time of *sampled A** (related to the number of sampled graphs it plans over before selecting the most frequent path) is limited to the time RAGS needed to find a path. We chose *A** to compare against as a simplified solution to the problem and *sampled A** because the method was previously introduced in a similar domain. The *greedy* approach provides a baseline for comparison to a purely reactive implementation.

The search algorithms were tested on a set of graphs generated with a uniform random distribution of 100 vertices over a space 100×100 in size and connected according to the PRM* radius [6]. Edge costs were represented by normal distributions with mean equal to the Euclidean distance between vertices plus an additional cost drawn from a uniform random distribution over $[0, 100]$. The variance of each distribution was drawn from a uniform distribution over $\left[0, \sigma^2_{max}\right]$, where $\sigma^2_{max} = \{5, 10, 35\}$ for the three separate sets of experiments. Note that a minimum cost of the Euclidean distance was enforced in the fol-



(a) (b)

Fig. 3. A sample search graph is shown in (a). The mean cost is the sum of the Euclidean distance plus a random additional cost. Edge variances are represented using greyscale on the graph. The darker the edges, the less variance there is on the cost. The non-dominated path set (red) and the executed RAGS path (blue) shown in (b) demonstrates the algorithm's ability to account for both the path cost distributions as well as the available path options to goal. Not only does it favor traversing edges with balanced mean costs and variances, it also maintains a high number of path options towards the goal.

lowing experiments. The start vertex was defined at $V_s = (0, 0)$, with the goal at $V_g = (100, 100)$. Figure 3a gives an example of a randomly generated graph, with edge variance shown in greyscale (darker lines having a smaller variance). In Figure 3b the non-dominated path set is shown in red, and the RAGS path is shown in blue.

In Figure 4 we show the results for 100 trials, where each trial drew new edge costs from the same distribution as described above. *Naïve A** performs well in the mean but is prone to outliers of more expensive paths, especially as the edge uncertainty increases due to edges with high variance along the path. RAGS is able to mitigate against such outliers by choosing safer routes with lower variances and more path options, demonstrating the benefit of risk-aware planning. Sampling all edges in the graph and planning iteratively, as in the *sampled A** approach, can account for variability in costs along the edges as long as it can sample enough paths. However this method is similarly prone to risky paths yielding high-cost outliers because it doesn't account for local information like RAGS. The performance of *greedy* search is prone to more variability in final costs as the edge cost variances increase. This trend can be attributed to the fact that the *greedy* algorithm performs search over the non-dominated path set, which becomes less representative of the true optimal path as the variances in the edge costs increase. At low cost variances, the non-dominated path set is a tighter representation of the optimal path set, and this bound becomes looser as cost variances increase. Due to the myopic nature of its planning, *greedy* is fallible to arriving at vertices with few path alternatives that all turn out to have high cost. RAGS does not suffer from the same performance decay since it accounts for the full path-to-goal cost distribution at each decision instance.

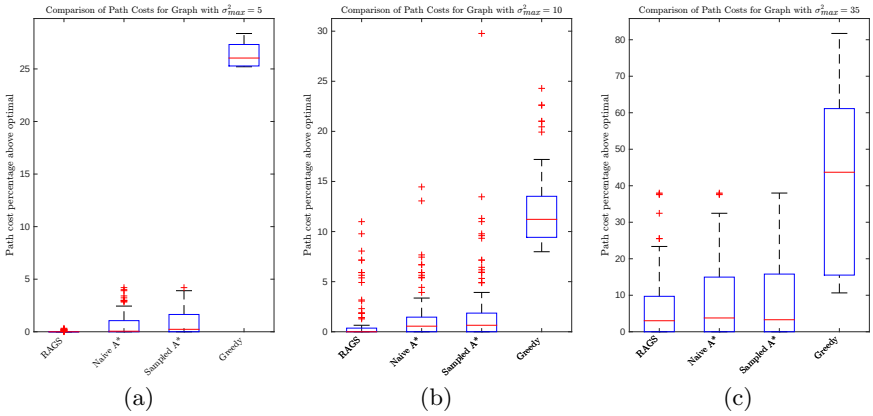


Fig. 4. Path search results over 100 samples of three graphs with uncertain edge costs; edge cost variances are drawn uniformly between 0 and $\{5, 10, 35\}$ for the three graphs (a)-(c), respectively. The difference in cost of the executed path and the true optimal path as a percentage of the optimal path cost is shown. Note that the true optimal path cost can only be calculated in hindsight.

The *greedy* results suggest that similar replanning strategies based only on the immediate edge cost updates may suffer from similar one-step lookahead myopia if downstream risks are not incorporated into the planning.

Table 1. Path Search and Execution Time (s)

σ^2_{max}	RAGS	A*	Greedy	Sampled A*
5	0.895 ± 0.150	0.102 ± 0.007	0.071 ± 0.007	109 Samples
10	$0.998 \pm .245$	$.111 \pm 0.008$	0.066 ± 0.006	111 Samples
20	$1.287 \pm .228$	$.108 \pm 0.008$	0.072 ± 0.006	130 Samples

(Sampled A* times are matched with RAGS)

The computation time averaged over 100 samples of 20 different graphs is presented in Table 1. Graph search and execution was calculated on a 3.2GHz Intel Core i7-4702HQ laptop. As expected, both $A*$ and *greedy* planning execute significantly faster than either RAGS or *sampled* $A*$. The initial $A*$ search is faster than any of the other tested algorithms since it returns only a single path which it then executes. Similarly, the *greedy* execution-time decisions require only the comparison of immediately neighboring edge costs. The main increase in computation time over $A*$ is due to the initial sweep for the non-dominated path set. Even so, this overhead is comparatively small when considering the size of the non-dominated set. On average, this set contained 69 paths and took 0.04s longer to compute than the $A*$ path.

As discussed in Section 4.1, the computation time of RAGS is heavily influenced by the pairwise risk comparison at each execution step and thus is sensitive to the branching factor of the non-dominated path set. Although this set is pruned as edge transitions are executed, causing path traversal decisions

to increasingly become faster to compute, the RAGS execution decision (3) actively attempts to maintain a large set of future path options. Despite this, these computation times suggest that real-time implementation on board a platform is feasible, and future work will investigate methods for improving the computational speeds of the algorithm.

6 Satellite Data Experiments

We also applied RAGS to a real world domain using satellite data. In robotic path planning there is often prior information available on the environment, but this information is not necessarily reliable. An example of this is overhead satellite imagery. In these trials, we used available satellite images, along with some filtering, to extract potential obstacles for a ground robot or a low-flying UAV. To convert the imagery into a useful mapping of obstacles, we performed a series of filters to provide a correspondence between obstacle density and pixel intensity. The satellite images were first converted to greyscale and were then blurred using a Gaussian filter. We then increased the contrast of the image. Finally, we eroded and then reconstructed the image to better identify trees and obstacles.

After the filtering process, the images provided a rough estimate of obstacles that could force the vehicle to slow down or fly around. In Figures 5-7, the brighter the pixel, the more likely there is to be an obstacle. The satellite information is too pixelated to provide fully reliable information, but we can use the imagery as an estimate of the obstacles in the environment. To do this we calculated the mean and variance of the pixel intensity values over an edge and used these to characterize the edge cost distributions. Similar to the previous comparison trials, the mean intensity was added to the Euclidean distance to provide a spatial scaling. Using the same method as before, we randomly sampled the space to generate a connected graph. The edges were assigned distributions, and then RAGS was used to search through the graph for a path from the top left start vertex to the bottom right goal vertex. Actual values of the edge costs were drawn from the distribution as the simulated robot moved through the terrain.

6.1 Results

We compared the performance of RAGS to the three other planning algorithms across 100 satellite images. The images are of fields with trees of varying tree densities and may also contain houses or other built structures. Images were captured at different resolutions as well as at different altitudes. The majority of the data have tree clusters scattered around the image to provide interesting path planning dilemmas. Three distinct environments are shown in Figures 5, 6, and 7. To avoid visual clutter, only the RAGS, $A*$ and hindsight optimal paths are shown here, compiled results for all four planning algorithms are presented in Figure 8.

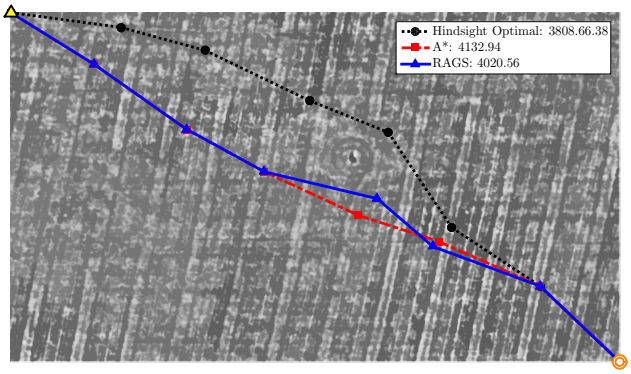


Fig. 5. The RAGS, $A*$, and the global optimal (known only in hindsight) paths are shown in the figure with the final path costs. The goal is to traverse from the top left start vertex to the bottom right goal vertex. The empty field is a test case showing that both algorithms plan direct paths as expected in an obstacle-free environment.

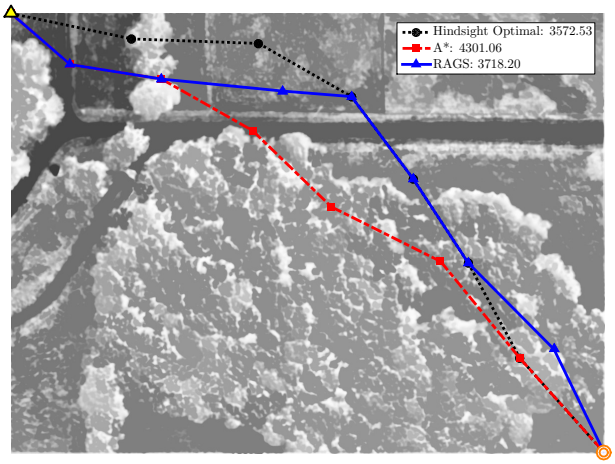


Fig. 6. Paths are planned through a dense cluster of trees surrounding the goal. Here we can see RAGS plans around the cluster, where there are more low cost paths, before traversing through a less cluttered area. RAGS takes advantage of the wide open region instead of searching for narrow tracks within the tree cluster.

In Figure 5 the paths through an empty field are straight from start (yellow triangle) to goal (orange circles). As expected there is little variance in edge costs, and the trajectories for RAGS and $A*$ are quite similar. Analyzing the paths found in Figure 6 is more interesting. Here we see the benefit of RAGS in obstacle-dense environments. The path from start to goal is blocked by a large cluster of semi-permeable forest. $A*$ executes a path through the center of the cluster that has a low cost in the mean but does not allow for easy deviations if the path is found to be untraversable. On the other hand, the path executed by RAGS demonstrates the nonmyopic nature of this algorithm. RAGS selects

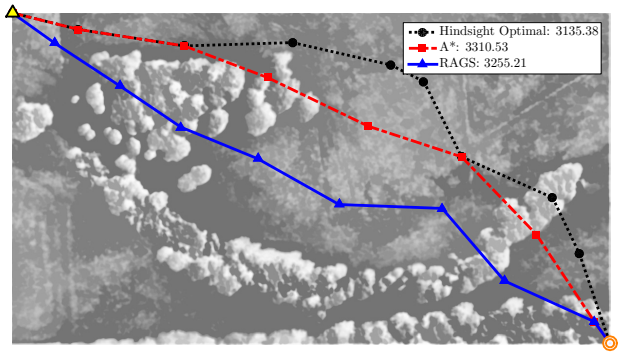


Fig. 7. The RAGS path is shown traveling straight through the initial cluster of trees to take full advantage of the direct route through the open field. $A*$ finds a path that does not navigate completely around the trees and so incurs additional traversal costs on top of taking a less direct route. The costs after execution show that RAGS is actually a cheaper path due to balancing the risk of finding a path through the initial tree cluster that connects to a more direct path to goal. This demonstrates the benefit of RAGS, knowing when to take risks and when to act conservatively.

a path that travels part way around the cluster to minimize the portion of the path within the dense section of the forest. Values are drawn from the edge distributions to calculate what would have been the optimal path in hindsight. The optimal path (known only after execution) is shown in black, and we can see that it follows a similar trajectory as RAGS.

The final test case can be seen in Figure 7, where a small cluster of trees stands between a direct path from the start to the goal. In blue, RAGS plans a path through the trees that is able to take advantage of the clearing in the center. In this example, RAGS is able to assess the risk of taking the shorter, more direct route through the trees and compare this to the expected cost of traveling around the cluster. In comparison, the $A*$ solution finds a less direct route to the goal; however it does not navigate completely around the trees and so incurs additional traversal costs on top of taking a longer path.

The compiled results for all tested algorithms are shown in a box plot of percent above the hindsight optimal in Figure 8. From the comparison on the three randomly generated graphs in Section 4.3, we showed that the relative performance of RAGS increases as edge variance increases. This is accounted for by the fact that the other planning algorithms are merely searching over the single heuristic of mean traversal cost. If variance is low then this can be enough to solve for a path that is close to the optimal solution. However, Figure 8 reveals that real world data sets can contain significant noise, and it is valuable to account for that variability during planning. This is especially evident in the *sampled* $A*$ result, which shows an overall lower mean cost but contains significant outliers (over twice as expensive) compared to the worst-case RAGS path.

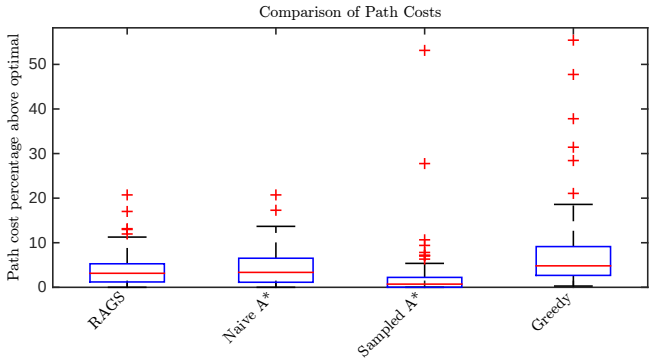


Fig. 8. Results from satellite data experiment, using 100 images. Box plots represent the path cost percent above what would have been in hindsight the optimal path. The same trends in performance are seen as with the simulated graphs. RAGS accounts for uncertainty in traversing the world, balancing risk of travel time (cost) to reduce expensive outliers seen in the comparative algorithms.

7 Conclusion

In this paper we have introduced a novel approach to incorporating and addressing uncertainty in planning problems. The proposed RAGS algorithm combines traditional deterministic search techniques with risk-aware planning. RAGS is able to trade off the number of future path options, as well as the mean and variance of the associated path cost distributions to make online edge traversal decisions that minimize the risk of executing a high-cost path. The algorithm was compared against existing graph search techniques on a set of graphs with randomly assigned edge costs, as well as over a set of graphs with traversability costs generated from satellite imagery data. In all cases, RAGS was shown to reduce the probability of executing high-cost paths over $A*$, *sampled* $A*$ and a *greedy* planning approach.

Our next step will be to implement the RAGS algorithm in an information optimization domain. In this case, edge cost probability distributions will represent environmental information, and paths will be generated based on the amount of information the vehicle may collect in different parts of the map. Additionally, we will investigate methods to reduce the computational complexity and memory requirements of the algorithm for implementation on a robotic platform.

Acknowledgements

This research has been funded in part by NASA grant NNX14AI10G and Office of Naval Research grant N00014-14-1-0509.

References

1. Polychronopoulos, G.H., Tsitsiklist, J.N.: Stochastic shortest path problems with recourse. Networks **27** (1996) 133–143

2. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2) (1968) 100–107
3. Koenig, S., Likhachev, M.: Fast replanning for navigation in unknown terrain. IEEE Transactions on Robotics **21**(3) (2005) 354–363
4. Hollinger, G.A., Pereira, A.A., Binney, J., Somers, T., Sukhatme, G.S.: Learning uncertainty in ocean current predictions for safe and reliable navigation of underwater vehicles. Journal of Field Robotics **33**(1) (2016) 47–66
5. Murphy, L., Newman, P.: Risky planning on probabilistic costmaps for path planning in outdoor environments. IEEE Transactions on Robotics **29**(2) (2013) 445–457
6. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. International Journal of Robotics Research **30**(7) (2011) 846–894
7. Latombe, J.C.: Robot Motion Planning. Springer Science & Business Media (2012)
8. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
9. Bohren, J., Rusu, R.B., Jones, E.G., Marder-Eppstein, E., Pantofaru, C., Wise, M., Mösenlechner, L., Meeussen, W., Holzer, S.: Towards autonomous robotic butlers: Lessons learned with the PR2. In: Proc. IEEE International Conference on Robotics and Automation. (2011) 5568–5575
10. Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of Fluids Engineering **82**(1) (1960) 35–45
11. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Robotics: Science and Systems. (2008)
12. Chaves, S.M., Walls, J.M., Galceran, E., Eustice, R.M.: Risk aversion in belief-space planning under measurement acquisition uncertainty. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems. (2015) 2079–2086
13. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: Proc. IEEE International Conference on Robotics and Automation. (2011) 723–730
14. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theoretical Computer Science **84**(1) (1991) 127–150
15. Ferguson, D., Stentz, A., Thrun, S.: PAO for planning with hidden state. In: Proc. IEEE International Conference on Robotics and Automation. Volume 3. (2004) 2840–2847
16. Monahan, G.E.: State of the Art—-A survey of partially observable Markov decision processes: Theory, models, and algorithms. Management Science **28**(1) (1982) 1–16
17. Hou, P., Yeoh, W., Varakantham, P.R.: Revisiting risk-sensitive MDPs: New algorithms and results. In: Proc. International Conference on Automated Planning and Scheduling. (2014)
18. Carpin, S., Chow, Y.L., Pavone, M.: Risk aversion in finite Markov Decision Processes using total cost criteria and average value at risk. In: Proc. IEEE International Conference on Robotics and Automation. (2016) 335–342
19. Marcus, S.I., Fernández-Gaucherand, E., Hernández-Hernandez, D., Coraluppi, S., Fard, P.: Risk sensitive Markov decision processes. In: Systems and Control in the Twenty-First Century. Springer (1997) 263–279
20. Sun, W., Patil, S., Alterovitz, R.: High-frequency replanning under uncertainty using parallel sampling-based motion planning. IEEE Transactions on Robotics **31**(1) (2015) 104–116
21. Feyzabadi, S., Carpin, S.: Risk-aware path planning using hierarchical constrained Markov Decision Processes. In: Proc. IEEE International Conference on Automation Science and Engineering. (2014) 297–303
22. Meliou, A., Krause, A., Guestrin, C., Hellerstein, J.M.: Nonmyopic informative path planning in spatio-temporal models. In: Proc. AAAI Conference. (2007) 602–607