Collision Posteriors on Graphs with Expensive-to-Evaluate Edges

Brian Hou, Sanjiban Choudhury, Gilwoo Lee, Matt Barnes, Siddhartha S. Srinivasa
Paul G. Allen School of Computer Science & Engineering, University of Washington
Email: {bhou, sanjibac, gilwoo, mbarnes, siddh}@cs.washington.edu

***Abstract*—Collision checking is a computational bottleneck in motion planning, requiring lazy algorithms that explicitly reason about when to perform this computation. Typically, such algorithms only check edges on the current shortest path to minimize the number of edges evaluated. However, they fail to exploit a key structure: the results of collision checking are highly correlated, and few checks can provide information about many configurations. We present strategies to learn and exploit collision correlations in order to reduce the checks required to compute the optimal collision-free path. On structured environments, leveraging these correlations reduces the number of collision-checked edges by an average of 11%.**

I. Introduction

Sampling-based strategies for motion planning operate by sampling from the continuous search space of all possible robot configurations to form a discrete graph (or roadmap), where vertices are robot configurations and edges represent simple movements between them [7]. To compute a collision-free path from an initial configuration to the goal, an algorithm must determine which edges of the graph the robot can traverse without colliding with obstacles in its environment. A common strategy for collision checking an edge is to discretize and check whether any configuration is in collision; edge checking is therefore typically one to two orders of magnitude more computationally expensive than configuration checking [6].

To mitigate this bottleneck, lazy motion planning algorithms defer collision evaluations between two configurations until that edge is potentially along the optimal path [3, 6, 5]. These planning algorithms maintain two sets of edges: *evaluated* edges (either VALID or INVALID) and *unevaluated* edges (optimistically assumed to be VALID). When deciding between several unevaluated candidate edges, edge-evaluation heuristics can focus collision checks on edges that have high probability of collision and quickly invalidate them.

Our key insight is that edge labels are highly correlated in real-world environments, enabling few edge evaluations to provide information about many edges. These correlations arise due to spatial distribution of obstacles or due to configuration-to-workspace mapping. Thus, edge-evaluation strategies should consider past edge evaluations as they decide which edges to evaluate in the future.

In this work, we examine a simple approach for learning collision posterior distributions from a dataset of previous planning problems. As edges are evaluated, these learned posteriors can continuously predict edge collision probabilities for the remaining unevaluated edges. Integrating these learned predictions with a lazy motion planner enables the algorithm to reason about which edges to prioritize during collision-checking. This collision posterior approach reduces the collision checks required to compute the optimal collision-free path by an average of 11%.

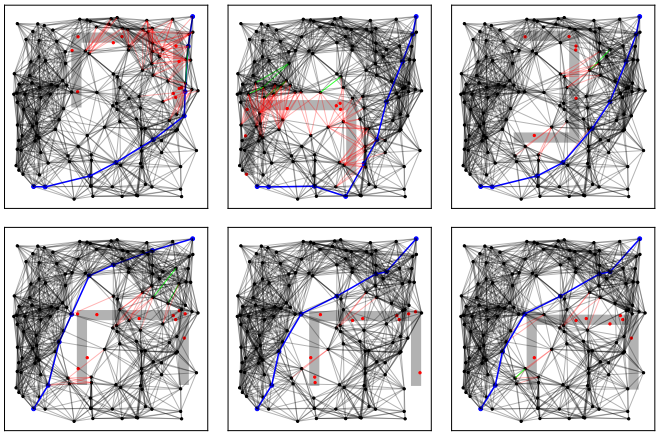

Fig. 1: Edges evaluated by LazySP using the proposed PosteriorFailFast selector. The algorithm returns the optimal path (blue) without evaluating most edges in the graph (black). Expensive collision checks have been focused on edges that are in collision (red), rather than edges that are collision-free (green). Obstacles (gray) are for visualization only.

II. Related Work

Planning with expensive collision-checking is a well-studied problem in motion planning. We focus on the paradigm of lazy motion planning, where an edge is only evaluated if it is potentially along the optimal path. Lazy-PRM* and Lazy-RRG* are asymptotically-optimal anytime lazy motion planning algorithms which only evaluate an edge if it would result in a shorter path [6]. The LazySP class of algorithms continuously re-solve the shortest path problem assuming that unevaluated edges are collision-free while lazily eliminating edges as they are evaluated to be in collision, replanning until the optimal collision-free path is found [3]. Lazy Receding Horizon A* balances the LazySP tradeoff between planning time and collision-checking time [8]. These algorithms describe several heuristics for edge evaluation, but typically do not reason about the correlation between edge evaluations. However, as we will discuss in Section IV, they can be

extended to this setting when there are examples of past planning problems to learn from.

Several machine learning approaches have been proposed to predict the validity of unevaluated edges. Esposito and Wright [4] argue that the adjacency matrix for a roadmap can be decomposed into the sum of a low-rank matrix and a sparse error matrix, and propose a convex optimization procedure to recover the unknown entries. This method predicts edge validities for a fully-connected graph from a proportionally small set of randomly observed entries, while we focus on predicting edge validities for a fixed graph that is much less connected. Pan et al. [9] use approximate nearest-neighbor queries to efficiently perform probabilistic collision checking for configurations and edges based on previously evaluated configurations. This approach only incorporates the validity of nearby points in configuration space to predict edge validities; however, points that are far apart in configuration space may still be informative, e.g. if they are close in task space. Furthermore, neither approach can learn from past planning problems to improve their predictions.

Many motion planning algorithms leverage edge collision probabilities to search more efficiently. Haghtalab et al. bound the number of edge evaluations of the LazySP class of algorithms in this probabilistic setting and prove that LazySP is asymptotically optimal [5]. POMP is an anytime motion planning algorithm that leverages edge collision probabilities to quickly find a collision-free path with few edge evaluations, then balances path length and path collision probability to find shorter paths [1]. The BISECT algorithm uses Bayesian active learning to find a feasible path from a library of paths, computing priors on edge collision probabilities from offline training examples [2].

III. Background: Lazy Shortest Path on Graphs with Expensive-to-Evaluate Edges

In the shortest path problem, we assume that we are given a graph $G = (V, E)$ and desired start and goal vertices $v_s, v_g \in V$. Traversing each edge e in the graph incurs a cost of $w(e)$. The optimal path from the start to goal $p^*(v_s, v_g)$ minimizes the total cost of edges in the path.

When edges in the graph are expensive to evaluate, we only assume knowledge of lower bounds for each edge weight $\hat{w}(e)$ in the set of edges E (e.g. Euclidean distance). The exact weight $w(e)$ is unknown until the edge is evaluated for collisions: a collision-free edge has cost $\hat{w}(e)$ and an edge in collision has infinite cost. Thus, the resulting shortest path in the graph with exact edge weights is collision-free. Since edge evaluations are expensive, we wish to minimize the number required to compute the shortest collision-free path.

We build on the LazySP framework proposed by Dellin and Srinivasa [3]. LazySP maintains an optimistic graph where an edge is assumed to be collision-free (i.e. edge weight is exactly $\hat{w}(e)$) until it has been evaluated to be in collision. It computes the shortest path on the optimistic graph, then selects an edge along this candidate path to evaluate according to an *edge-selector* function. If an edge along the candidate path is evaluated to be in collision, a new candidate is proposed based on the updated optimistic graph. If all edges along a candidate path are evaluated to be collision-free, then it must be the optimal path $p^*(v_s, v_g)$. LazySP assumes that each edge evaluation takes the same time and edge evaluations dominate computation time relative to solving shortest path queries. The choice of edge selector has a drastic effect on the number of edges that LazySP evaluates.

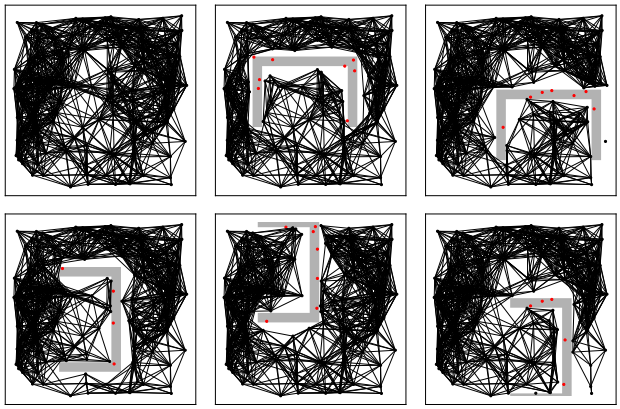

Fig. 2: Training examples from the BugTrap environment distribution. The top-left figure is the graph that is shared by all environments in the distribution. Edges that are in collision with the BugTrap obstacle (gray) have been omitted.

IV. Planning with Collision Posteriors

Let $P(\phi)$ be the distribution of environments that the planner will be tasked with solving. Our objective is to compute an edge-selector function for this distribution that minimizes the number of edges evaluated as LazySP computes the shortest path. Although we do not have direct access to $P(\phi)$, there is a dataset of samples $\{\phi^i\}$ to learn from. We divide this into two subtasks: learning edge collision posteriors from example environments and leveraging those approximate posteriors to inform edge evaluation.

A. Learning Edge Collision Posteriors

In our dataset, each distribution over environments $P(\phi)$ is associated with a single explicit graph $G = (V, E)$. For each environment in the dataset, we precompute the collision statuses for every edge in the graph; different subsets of edges will be in collision depending on the arrangement of obstacles in each environment.

We model the validity of each edge as a Bernoulli random variable E_i. Each environment ϕ assigns a binary value (either in collision or collision-free) to each E_i. Our goal is to estimate the posterior distribution $P(E \setminus E_{eval} | E_{eval})$, where E_{eval} is the set of evaluated edges. In this work, we use the Naïve Bayes assumption that the posterior can be factored as

$$P(E_i = e_i | E_{eval}) \propto P(E_i = e_i) \prod_{e_j \in E_{eval}} P(E_j = e_j | E_i = e_i)$$

B. Edge Collision Posterior Selectors

Given a set of evaluated edges and a candidate shortest path, a selector can compute the posterior probability that each unevaluated edge is in collision, then select an edge to evaluate accordingly. In this work, we adopt a fail-fast strategy of selecting the edge with the highest posterior probability of being in collision. By focusing on edges with high collision probability, our PosteriorFailfast selector quickly invalidates candidate paths.

V. Experiments

We evaluate these strategies on a variety of 2D motion-planning problems [2]. We focus on these simply for ease of visualization; the approaches we propose depend on the number of edges in the graph, and do not have an explicit dependence on the dimension of the configuration space. Figure 2 shows examples from the BugTrap dataset.

We compare three LazySP edge selectors. Forward selects the unevaluated edge that is closest to the start of the candidate path [3]. PriorFailfast selects the edge with the highest prior probability of collision [2]. Our proposed selector, PosteriorFailfast, predicts the posterior distribution with Naïve Bayes and selects the edge with the highest posterior probability of collision. PriorFailfast and PosteriorFailfast leverage the same dataset (800 environments) to estimate collision probabilities.

Before initiating lazy motion planning (and edge evaluation) on the graph, vertices are eagerly evaluated for collisions. If a vertex is in collision, then each edge incident on that vertex is also marked as in collision. These observed edge statuses are part of the initial problem state, and can be leveraged by PosteriorFailfast when it predicts the remaining edges.

Figure 3 compares the PosteriorFailfast selector directly with the other two selectors on the test set (200 environments). Each point corresponds to one environment, where the coordinates are the number of edges evaluated by each selector. Points are colored according to the selector which evaluated fewer edges on each problem; green points indicate problems on which PosteriorFailfast outperforms Forward and PriorFailfast, respectively.

PosteriorFailfast outperforms PriorFailfast on 88% of planning problems, evaluating an average of 11% fewer edges. The number of edges evaluated by PosteriorFailfast ranges from 46% to 116% of the edges that PriorFailfast evaluates (Figure 4). Although PosteriorFailfast may not maximize the usefulness of the collision posterior, this strategy does not seem to evaluate many more edges when it performs worse than PriorFailfast.

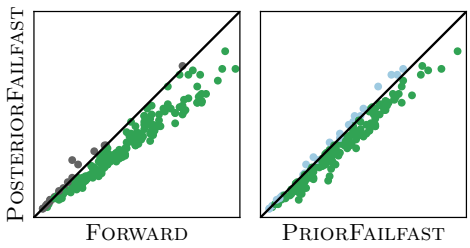

Fig. 3: Pairwise selector comparison on BugTrap environments. Each point corresponds to a planning problem, where the (x, y)-coordinates are the number of edges evaluated with that selector. Points are colored by which selector evaluated fewer edges on each problem: PosteriorFailfast (ours, green), PriorFailfast (blue), or Forward (gray).

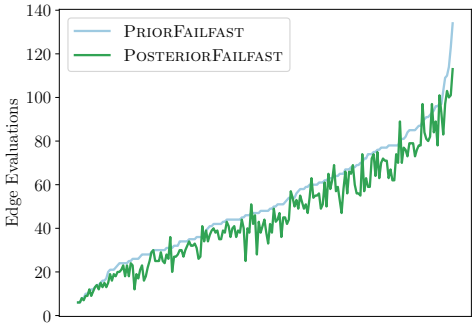

Fig. 4: Edge evaluation results for LazySP on BugTrap environments (lower is better). Results are ordered by increasing number of edge evaluations performed by the PriorFailfast selector as a proxy for problem difficulty.

VI. Discussion and Future Work

We have presented an approach for learning an edge collision posterior model from data and combining that model with a lazy motion planning algorithm. We have relied on the completeness of the planning algorithm while boosting performance on average using learning. The fail-fast edge selectors that we have considered in this work are effective, but they suffer from the same myopia of concentrating on the current candidate path. Reasoning about future candidate paths that are also being eliminated may further narrow the gap to a Bayes-optimal edge selector.

We believe that this approach can scale effectively to higher-dimensional configuration spaces since the models are dependent on the number of edges in the graph, rather than the dimensionality of the configuration space. Validating this hypothesis on more challenging problems (e.g., manipulators) is a major direction of our future work.

While we have focused on the single-source shortest path problem, we believe that these posteriors will also be valuable for other motion planning problems (e.g., near-optimal path planning, feasible path identification). In future work, we hope to explore some of these problems.

Acknowledgements

Brian Hou is partially supported by NASA Space Technology Research Fellowships (NSTRF). Gilwoo Lee is partially supported by Kwanjeong Educational Foundation. This work was partially funded by the National Institute of Health R01 (#R01EB019335), National Science Foundation CPS (#1544797), National Science Foundation NRI (#1637748), the Office of Naval Research, the RCTA, Amazon, and Honda.

References

[1] S. Choudhury, C. Dellin, and S.S. Srinivasa. Pareto-optimal search over configuration space beliefs for anytime motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[2] S. Choudhury, S. Javdani, S.S. Srinivasa, and S. Scherer. Near-optimal edge evaluation in explicit generalized binomial graphs. In *Advances in Neural Information Processing Systems*, 2017.

[3] C. Dellin and S.S. Srinivasa. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *International Conference on Automated Planning and Scheduling*, 2016.

[4] J.M. Esposito and J.N. Wright. Matrix completion as a post-processing technique for probabilistic roadmaps. In *Workshop on the Algorithmic Foundations of Robotics*, 2016.

[5] N. Haghtalab, S. Mackenzie, A.D. Procaccia, O Salzman, and S.S. Srinivasa. The Provable Virtue of Laziness in Motion Planning. In *International Conference on Automated Planning and Scheduling*, 2018.

[6] K. Hauser. Lazy collision checking in asymptotically-optimal motion planning. In *IEEE International Conference on Robotics and Automation*, 2015.

[7] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[8] A. Mandalika, O. Salzman, and S.S. Srinivasa. Lazy Receding Horizon A* for Efficient Path Planning in Graphs with Expensive-to-Evaluate Edges. In *International Conference on Automated Planning and Scheduling*, 2018.

[9] J. Pan, S. Chitta, and D. Manocha. Faster sample-based motion planning using instance-based learning. In *Workshop on the Algorithmic Foundations of Robotics*, 2012.