On the Impact of Uncertainty for Path Planning

Jérôme Guzzi, R. Omar Chavez-Garcia, Luca M. Gambardella, Alessandro Giusti

***Abstract*— We consider the problem of planning paths on graphs with some edges whose traversability is uncertain; for each uncertain edge, we are given a probability of being traversable (e.g., by a learned classifier). We categorize different interpretations of the problem that are meaningful for mobile robots navigating partially-known environments, each of which yields a different formalization; we then focus on the case in which the true traversability of an edge is revealed only when the agent visits one of its endpoints (*Canadian Traveller Problem*). In this context, we design a large simulation campaign on synthetic and real-world maps to study the impact of two different factors: the planning strategy, and the amount of uncertainty (which could depend on the quality of the classifier producing traversability estimates).**

I. Introduction

Path planning on graphs is a key problem for mobile robot navigation. In most applications, edges in graphs have one associated cost, and standard algorithms are applied to find the lowest cost path. Such cost may be the distance, time needed to traverse the edge, energy, etc. In all cases, one assumes that each edge can be traversed. Given a graph, there is one solution, which can then be executed by the robot; in case new information becomes available during the execution, the robot can re-plan its path to the target.

In this paper we consider the case in which some *hidden edges*, in addition to a deterministic cost, carry probabilistic information about whether that arc is in fact traversable. This is a very relevant scenario in many situations in which knowledge about the environment is uncertain (e.g., when we do not know whether a door is open or not). Even with perfect knowledge of the environment, one may still be unsure whether a robot can successfully traverse an edge, e.g., a ground robot dealing with a stretch of uneven terrain at the limit of its abilities (in this case, the traversal probability can be estimated with a learned classifier [1], [2]).

Properly representing and handling such probabilistic information is fundamental for autonomous robots that operate in real-world, unstructured environments. Humans routinely do this in many situations too: for example, when walking off-path in natural environments, when caught walking with unsuitable shoes on city streets full of puddles, or when a wheelchair-bound person has to find their way around complicated architectures with uncertain accessibility.

Depending on the scenario, the fact that a hidden edge is non-traversable has different interpretations: e.g., if a door is not open, the robot will notice that when approaching, and will be able to plan an alternative path; if an uneven terrain is not passable, the robot may remain stuck on it and be unable to proceed or backtrack. Even though these differences may seem subtle, they yield very different *formalizations* of the path-planning problem. In Section II we categorize such interpretations, and discuss related literature. Then we focus on one specific interpretation which is very relevant for robotics applications, which assumes that it is always possible to plan an alternative path (the *Canadian Traveller Problem*), and which we formalize in Section III-A.

We assume that a non-ideal estimator uses pre-existing knowledge (e.g., from an aerial image, 3D reconstruction, geographic information system) to produce probabilistic traversability information for each hidden edge (see Fig. 1 and Section III-B)). For example, the classifier may know that there is rugged terrain in a given area and therefore assign a 50% probability for the corresponding edge to be traversable by the robot; or the estimator may use existing information to estimate that a door may be open with a 90% probability. Uncertainty has two sources: 1) the inaccuracy of the estimator (which is not always correct *and* confident); 2) the number of [hidden] edges that require traversability estimation. In this context, the *optimal policy* selects the robot's actions also taking into account the risk and cost of backtracking if a non-traversable edge will be discovered; we discuss such policy and simpler strategies in Section III-C.

How much does the performance of these strategies depend on the amount of uncertainty? The **main contribution** of this work focus on answering this question. We setup a large simulation campaign on synthetically-generated and real-world maps (Section IV), report results (Section V) and discuss them (Section VI); Section VII concludes the paper.

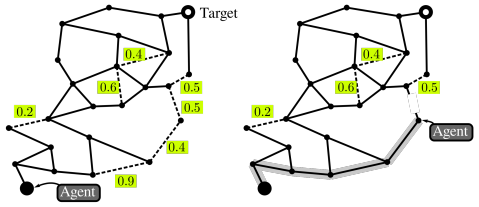


Fig. 1: Left: we consider an agent on a graph, where some edges are hidden (dashed lines). For each hidden edge, a non-ideal estimator provides a calibrated estimate for the probability (light green background) that it is traversable. When visiting a node, the agent has revealed to it the true traversability of all incident edges, and replans accordingly. Right: faced with a non-traversable edge, the agent has to backtrack to reach the target through a different path.

This research was supported by the Swiss National Science Foundation through the National Center of Competence in Research (NCCR) Robotics.

All authors are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, Switzerland. jerome@idsia.ch

Results and code to replicate our experiments are available at https://github.com/jeguzzi/resilience

II. Related Work

A. Interpretations of traversability for planning problems

We consider the case of path planning on graphs whose edges have an associated traversal probability. Once the robot is at a node n, incident to edge e that has traversal probability $q(e)$, it may turn out that e is in fact non-traversable. Three alternative interpretations for this event are meaningful in robotics.

1) The robot gets stuck while attempting to traverse e, i.e., it can not proceed nor backtrack.
2) The robot tries to traverse e but fails, hence remains at n (and possibly pays a cost). Attempting to cross e again may work, with the same probability $q(e)$.
3) The robot observes that e is non-traversable, and attempting to traverse it makes no sense. e is removed from the graph and the robot has to plan an alternative path to its target.

The first scenario gives rise to a multi-objective optimization problem. Each path from source to target will be associated to a total cost and to a survival probability. Assuming that the traversability of each edge is independent, the total traversal probability will be the product of the traversal probability of each edge on the path. The ideal path has a low cost and a high traversal probability: these two objectives can be handled with standard multi-objective optimization techniques [3], which have been applied to path planning for vehicles on uneven ground [2], [4].

In the second scenario, one may retry the edge until successfully traversed; this yields a shortest path planning problem with stochastic costs, which is a widely researched variation [5] with many real-world applications.

The third scenario is the focus of our paper. It is known as the Canadian Traveller Problem [6].

B. The Canadian Traveller Problem

Formally, the Canadian Traveller Problem (CTP) searches for the optimal policy to navigate a graph with some *hidden edges* of unknown traversability: information that will be revealed only upon arrival at an incident node.

In robotics, we are confronted with the same problem when a robot can decide whether an edge (whose traversability was previously uncertain) is traversable by using local sensing. For example, when the robot arrives near an area marked on its map with uncertain traversability, sensing can reveal information such as: the door is closed; the grass is too wet and slippery; the slope is too steep to climb.

Computing a policy for the CTP that minimizes the expected path cost is a #P-hard problem [7]. If we frame the problem as a partially observable Markov decision process (POMDP) and use Value Iteration to compute the optimal policy, computation cost growths $o(3^n)$ with the number n of hidden edges. This makes it practically impossible to solve CTP for more than a few tens of hidden edges, except for particular types of graphs, like DAGs, for which an exact policy have been explicitly formulated [8].

On complex POMDPs, the optimal policy can be efficiently approximated offline [9] (i.e., the policy is computed only once) or online [10] (i.e., the policy is revised during execution); heuristics specific to the CTP are based on Monte Carlo Sampling [11], [12] and AND-OR trees [13].

Extensions of CTP account for remote sensing [14] and multi-agent systems [15]. Adjusting traversability beliefs along the path, according to Gaussian Processes, allows one to model realistic problem, where uncertainty on different edges is not independent [16].

In this paper, contrary to the mentioned related works, we do not investigate how to efficiently compute or approximate the optimal policy for CTP. Instead, we study the impact of uncertain estimations on the *quality* of policies.

III. Model

A. Problem formulation

We are given a graph $G = (N, E)$ and an agent that moves on G. At the beginning, the traversability $r : H \to \{0, 1\}$ (0: non-traversable, 1: traversable) of some edges $H \subseteq E$ is not known to the agent. A non-ideal estimator estimates the traversal probability as $q : H \to [0, 1]$. The agent uses knowledge of q to navigate between a starting node $s \in N$ and a target node $t \in N$ according to navigation policy π. We assume that s and t are connected.

The problem is framed as a POMDP, whose states (n, k) are given by a node $n \in N$ and by the current knowledge $k : H \to \Omega = \{0, 1, \text{unexplored}\}$ of the true values of traversability of H. A navigation policy $\pi : N \times \Omega^H \to N$ selects the next node the agent will travel to according to its state. When the agent reaches a new node n, it observes the true value of some edges $H_n \subseteq H$ and sets $k(e) = r(e), \forall e \in H_n$. In the original CTP, H_n is the set of edges incident to n. The cost of a policy is the cost of the trajectory from s to t that it generates. The optimal policy π_{opt} is defined as the policy with the lowest expected cost.

If the traversability estimation q is exact, it associates a 100% traversal probability to hidden edges that are in fact traversable, and a 0% probability to edges that are not; then we can expect that π_{opt} will lead the agent to follow the minimum-cost path.

However, in the following we assume q to be inexact, i.e., it may assign non-zero probabilities to edges that are in fact non-traversable. Then, the optimal policy may lead the agent to reach a node where an edge, that would be followed next, is revealed to be non-traversable, and thereafter backtrack to follow a different path. Similarly, if a traversable hidden edge on the minimum-cost path is assigned a probability less than 100%, the optimal policy may prefer a longer path.

We are interested in how the cost of a policy depends on the quality of the estimations, which we assume are generated by a binary classifier.

B. Binary traversability classifier

We model q as a stochastic function; given an edge with a true traversability r, it assigns a traversability probability by sampling from a probability distribution that depends on

r (Fig. 2). To keep the model simple, we assume that the distribution is symmetric if we exchange the classes; this models the assumption that the classifier works equally well for traversable and non-traversable edges

$$p(q|r=0) = p(1-q|r=1) = B_{\alpha,\beta}(q) = B_{\beta,\alpha}(1-q). \quad (1)$$

$B_{\alpha,\beta}$ represents the family of Beta distributions, which is well suited to model binary classifiers [17], [18], [19], [20].

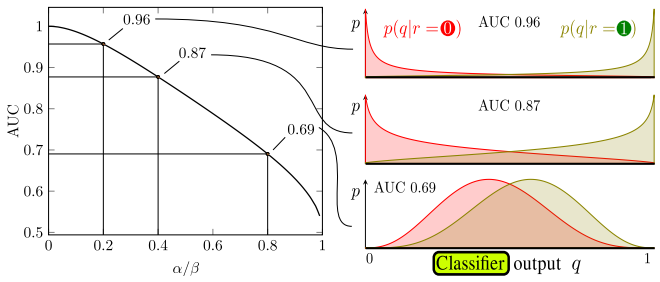


Fig. 2: Left: relation between α, β and AUC for a calibrated classifier ($\beta = \alpha + 1$). Right: probability distribution of the classifier output $q(e)$ applied to a non-traversable (red) or traversable (green) edge; for a strong (top), medium-quality (middle), weak (bottom) classifier. Note that the strong classifier returns polarized (close to 0.0 or 1.0) outputs, whereas the weak classifier is uncertain and aware of it (outputs are close to 0.5)

We limit our analysis to *calibrated* classifiers (i.e., $p(r = 1|q) = q$), which is the case if $\beta = \alpha + 1$: this implies that the score returned by a calibrated classifier can be interpreted directly as a probability by the agent. Intuitively, this means that if we collect many hidden edges for which the classifier returned a given probability q, a fraction close to q of them will in fact be traversable.

A calibrated classifier may or may not be accurate. Following the best practices in Machine Learning [20], we measure a classifier's quality via its Area Under the Receiver Operating Characteristic Curve (AUC). AUC values range from 0.5 (for a classifier that returns random or constant answers) to 1.0 (for an ideal classifier). Let e_0 be a random non-traversable edge ($r(e_0) = 0$), and e_1 be a random traversable edge ($r(e_1) = 1$). The AUC value $q_{\text{auc}}$of classifier q can be intuitively interpreted as the probability that $q(e_1) > q(e_0)$. The classifier returning exact answers (1.0 for traversable edges, 0.0 for non-traversable edges) is calibrated and has $q_{\text{auc}} = 1.0$.[1] A classifier returning always 0.5 is also calibrated (on a balanced dataset) but has $q_{\text{auc}} = 0.5$, which means that its answers are not informative.

For any given AUC value $0.5 \leq q_{\text{auc}} \leq 1.0$, there is a single choice of the pair $\alpha(q_{\text{auc}}), \beta(q_{\text{auc}})$ that yields a calibrated classifier (see Fig. 2).

C. Optimal and baseline policies

The optimal policy π_{opt} is defined as the policy with the lowest *expected* cost: for a given state, the action is chosen by accounting for the cost and likelihood of all possible realizations; in practice, the optimal policy will account for the risk and cost of backtracking when deciding to follow a path that traverses an hidden edge; we compute such policy using Value Iteration [13].

We compare π_{opt} with a family of baseline policies $\pi(\tau)$, parametrized by a threshold $\tau \in [0, 1]$. A policy $\pi(\tau)$ is defined as follows. In any state (n, k), we consider the subgraph composed by all and only the edges which are either known to be traversable, or unexplored, with $q(e) \geq \tau$ (this excludes the edges known to be non-traversable and unexplored edges with $q(e) < \tau$). If at least one path to the target exists in such subgraph, the shortest is computed and its first edge is traversed; else, the path with the highest probability of being traversable is computed (regardless of cost) on the full graph, and its first edge is traversed.

For any τ, $\pi(\tau)$ is guaranteed to eventually lead an agent to its target t as long as t is reachable from the source node s. $\pi(\tau)$ defines *reactive* policies which decide which edge to traverse next by making hard assumptions on the traversability of all unobserved edges, but revise these decisions as soon as new edges are observed.

$\pi(0)$ is a baseline *optimistic* policy that strives for the shortest path, ignoring classifier estimations and assuming all unobserved edges are traversable.

$\pi(1)$ is a baseline *pessimistic* policy: it assumes that hidden edges are non-traversable unless observed to be traversable. In the (common) case in which this does not yield a path to the target, this policy always chooses the action which proceeds along the path with highest traversal probability, ignoring edge costs.

IV. Experimental setup

In the following, we generate many planning problem instances, and compare the performance of different policies.

One instance is defined as follows (see right of Figure 3).

- We consider: a graph $G = (N, E)$, in which each edge has an associated cost; a pair of source and target nodes $s, t \in N$; a set H of hidden edges $H \subseteq E$.
- We generate one realization $r : H \to \{0, 1\}$ of the hidden edges true traversability, unknown to the agent; it assigns a binary traversability value (traversable or non-traversable) to each hidden edge. Each $r(e)$ is *independently* generated following a Bernoulli(0.5) distribution; if in the resulting graph t is not reachable from s, a new realization is drawn.
- We generate one realization $q : H \to [0, 1]$ of the traversal probabilities, which are provided to the agent. q is generated according to r by a classifier with a given AUC value q_{auc}. In particular, we sample the classifier output from $q(e) \sim B_{\alpha(q_{\text{auc}}),\beta(q_{\text{auc}})}$ if $r(e) = 0$ or from $q(e) \sim B_{\beta(q_{\text{auc}}),\alpha(q_{\text{auc}})}$ if $r(e) = 1$ (see Figure 2).

Once an instance is defined, we compute the optimal policy π_{opt}, simulate an agent following it in the realization r, and measure the cost of the resulting trajectory. We do the same with baseline policies $\pi(\tau)$ for different values of τ.

[1] $q_{\text{auc}} = 1.0$ does not imply calibration: the classifier returning 0.9 for all traversable edges and 0.1 for non-traversable edges is *not* calibrated but has $q_{\text{auc}} = 1.0$ and perfect accuracy.

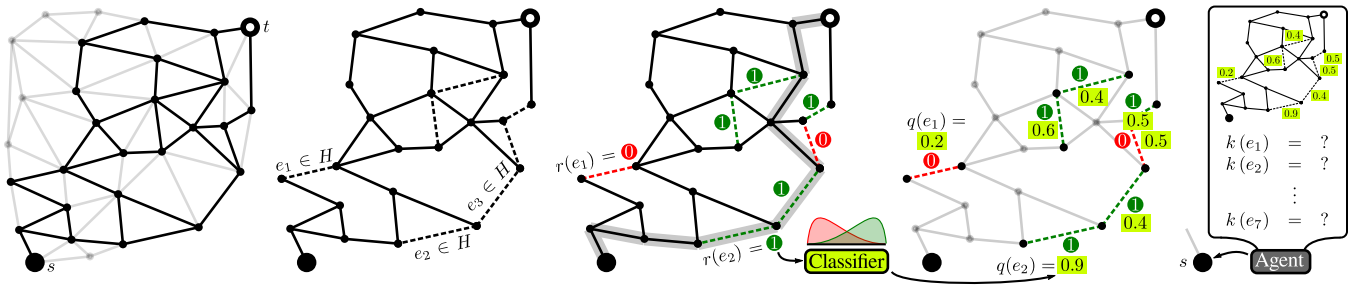


Fig. 3: Instance generation with a random graph, from left to right: generation of nodes and edges; random choice of edges in H; realization of each edge as traversable (dark green) or non-traversable (red) (ensuring connectivity between s and t, gray highlighted path); realization of the traversability estimates (light green) according to the classifier's model; the agent at the beginning of the simulation at node s knows the graph and q, but not r.

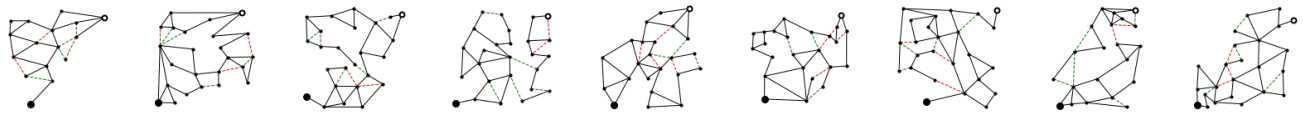


Fig. 4: 9 out of 100K random graphs and realizations with 7 hidden edges (with an average of 23 nodes and 35 edges). On each graph, we apply classifiers of different quality to generate more than 1M planning instances.

We do not report these costs directly; instead, we are interested in the ratio $c(\pi) \geq 1$, called *competitive ratio*, between such costs and the cost of the minimal-cost path in G (which can be computed given r).

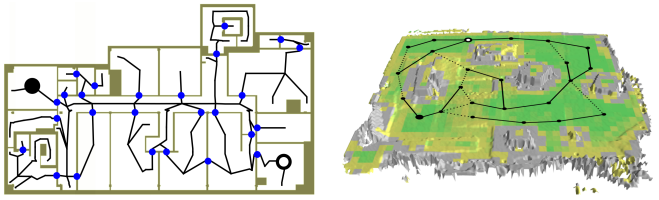
In each of the experiments below, we analyze the effect of a different parameter (q_{auc}, $|H|$, τ) and report statistics about competitive ratios of each policy over many instances.

A. Random graphs

We generate random graphs as illustrated in the first three illustrations (from left to right) of Fig. 3 by: (1) drawing 30 points uniformly between $[0, 1] \times [0, 1]$; (2) connecting the points using a Delaunay triangulation; (3) selecting s and t at the bottom-left and top-right corner respectively; (4) randomly deleting half of the edges without disconnecting s and t; (5) selecting H as a random subset of edges with a predefined cardinality; and (6) randomly selecting one realization r in which s and t are connected. We use simple strategies to make sure we generate interesting planning instances. For example: in (4) we prune parts of the graph that no policy would visit (i.e., branches that lead nowhere); in (5) we force that at least one hidden edge separates s and t along the minimal-cost path but we avoid picking hidden edges that would anyway need to be passed (e.g., bridges between s and t). For each experiment, we generate planning instances for 100K random graphs with a realization r (see Fig. 4).

B. Indoor map

We use a floor map of a real building where an agent has to estimate the probability that doors are open (i.e., traversable). The navigation graph depicted in Fig. 5a is composed by local trajectories derived from the building geometry and has 187 nodes and 236 edges; s and t are located in two rooms at the opposite side of the building . The set of H hidden edges contains all 9 doors that may be traversed when traveling from s to t. We run simulations over all ($2^9 = 512$) realizations r, generating 100 instances for each classifier q_{auc} value (for total 500K instances).



(a) Indoor floor map: navigation graph with doors that may be locked (blue circles), source (black circle) and target (white circle) nodes.

(b) Rugged terrain map with classifier outputs [2] for traversable (green), non-traversable (gray) and uncertain (yellow) terrain.

Fig. 5: Real-world navigation graphs between source (black circle) and target (white circle).

C. Rugged terrain map

Figure 5b shows a 3D map from the ETH-ASL traversability dataset [21]: an experimental scenario with several obstacles, such as bumps, ramps, holes, boxes and slippery surfaces. A traversability classifier predicts whether the robot will be able to traverse a patch of terrain [2]. We draw the navigation graph by hand with 24 nodes and 30 edges. We take into account traversability estimations to label edges as traversable, non-traversable or with uncertain traversability. We identify a total of 8 uncertain edges (corresponding to challenging terrain such as ramps, boxes edges or high bumps) which are modeled as hidden edges in H. As above, we use a single graph from which we generate all ($2^8 = 256$) realizations r, generating 100 instances for each classifier AUC value (for a total 250K instances).

V. Experimental Results

In Section V-A, we analyze the performance of policies on random graphs with 7 hidden edges. Then, we look at results on random graphs with different numbers of hidden edges (Section V-B) and on real-world maps (Section V-C).

A. Random graph with $|H| = 7$

1) Baseline policies: Figure 6a illustrates the effect of different values of τ on the average performance of baseline policies $\pi(\tau)$. As expected, the performance of the optimistic policy $\pi(0)$, which ignores classifier outputs, does not depend on the classifier quality; moreover it has the lowest average performance. Policies with $\tau \approx 0.6$ perform best regardless of classifier quality. For low-quality classifiers, the performance penalty for being too pessimistic (high τ) is progressively reduced, to the point that, for extremely low-quality classifiers, the pessimistic policy $\pi(1)$ offers comparable performance. More precisely, from a smooth u-shaped function for high q_{auc}, the dependency of the competitive ratio on τ becomes, as we approach $q_{\text{auc}}= 0.5$, a step-like function that just discriminates if $\tau > 0.5$.

For all other experiments, we only report results for $\pi(0)$ and $\pi(1)$, which have opposite distinguishing characteristics and are clearly interpretable, and for the nearly optimal choice of $\tau = 1/2$. In fact, $\pi(1/2)$ has also a simple interpretation: it considers as traversable any edge that has a higher probability to be traversable than not.

2) Impact of classifier quality on competitive ratio distribution: Figures 6c-f report mean, 95%-quantile and cumulative distribution of competitive ratios versus classifier quality. As expected, the competitive ratios are close to 1 when the classifier is very accurate ($q_{\text{auc}}\approx 1$) and grows as the classifier quality decreases. The optimal policy π_{opt} is almost always better than any baseline policy. The only exception is the policy $\pi(1/2)$, which has a higher probability of minimal cost ($c = 1$). Not surprisingly, π_{opt} has the lowest mean cost. For accurate classifiers, policy $\pi(1/2)$ is on par with the optimal policy, but performs significantly worse for inaccurate classifiers. In fact, for inaccurate classifiers, the gap between π_{opt} and $\pi(1/2)$ increases while the gap between π_{opt} and $\pi(0)$ or $\pi(1)$ decreases.

In general, the impact of the classifier quality on the average competitive ratio is limited ($\overline{c(\pi)} \leq 1.11$). For reference, the highest competitive ratio sample over all planning instances and policies is 5.23. When we consider the 95%-quantile, the impact is more significant, but the relations between the four policies remain similar.

3) Policies comparison on planning instances: Figure 6b illustrates the probability that a policy ranks first (possibly tied) among other policies on a random instance; this represents the fraction of samples for which the agent will not regret (in the aftermath) having followed a particular policy. For a good part of these instances, the agent actually follows the shortest path. We note that the gap between all four policies fades for low-quality classifiers, while for high-quality classifiers π_{opt} and $\pi(1/2)$ finds the best path much more often than $\pi(0)$ or $\pi(1)$.

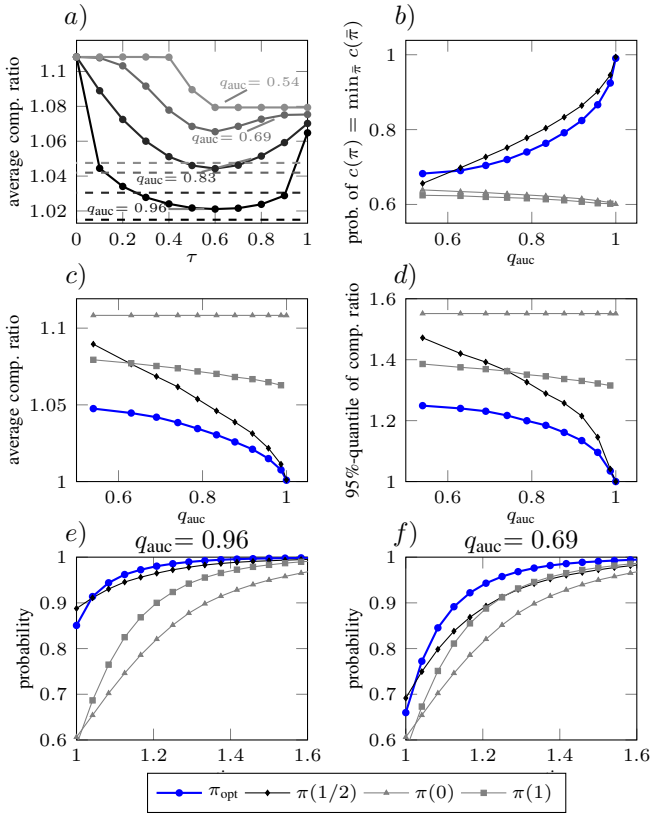


Fig. 6: Competitive ratio of policies on 100K random graphs with 7 hidden edges. (*a*): average as a function of τ, with high-quality (black), medium-quality (dark gray) and low-quality (light gray) classifiers; performance of π_{opt} is represented by dashed lines of the same color. (*b*): probability that a policy has at most a cost equal to any other policy versus q_{auc}. Average (*c*) and 95%-quantile (*d*) versus q_{auc}. Cumulative distribution of competitive ratios for high-quality (*e*) and low-quality classifiers (*f*).

B. Random graph with $1 \leq |H| \leq 9$

We report in Fig. 7 how the competitive ratio increases as a function of the number of hidden edges $|H|$. The impact of the classifier quality on the policies' performance growths with planning complexity (higher $|H|$), yet the relative performance between policies remains similar.

We also report the measured mean computational cost of the optimal policy π_{opt} by Value Iteration using a single modern CPU core. The baseline policies $\pi(\tau)$ have negligible computation costs that scale polynomially with $|H|$.

C. Real-world graphs

We compare the results on random graphs with experiments on two real maps, where we study the impact of classifier quality on the policies' performance. Note that although we sample over all possible realizations (and many classifications), these two maps represent just two graph instances and the policies performance has a large variability over different graphs.

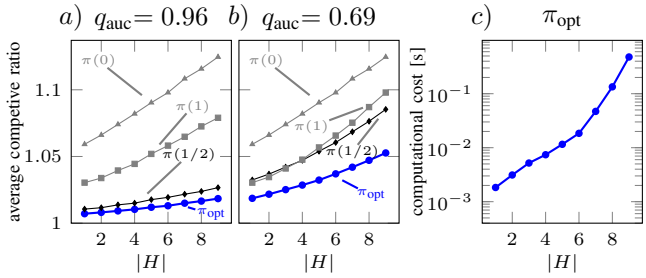


Fig. 7: Competitive ratio on random graphs as a function of $|H|$, with a high-quality (*a*) and a low-quality (*b*) classifier. (*c*): mean computational cost for a single instance of π_{opt}.

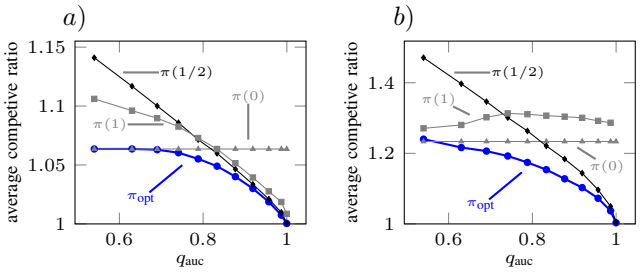


Fig. 8: Average competitive ratio of policies as a function of classifier quality on real world maps: Indoor floor map (*a*) and Rugged terrain map (*b*).

a) Indoor map: Figure 8a summarizes the policies' performance on a real building map, whose hidden edges correspond to doors (that may be open or closed). The impact of estimation quality on the optimal policy is comparable to the previous results. Interestingly, on this particular map, baseline policy $\pi(0)$ perform much better (and $\pi(1/2)$ much worse) than on most random graphs.

b) Rugged terrain map: We report a similar experiment for the Rugged terrain map in Fig. 8b. The average competitive ratios for this graph are high when compared to random graphs ($\geq 90\%$-quantile), denoting an harder than average planning instance. As above, the performance of $\pi(0)$ with respect to π_{opt} is much better than most random graphs.

VI. Discussion

The following interesting aspects summarize our findings.

a) Lower estimation quality makes the planning problem harder: The gap between the average performances of the best heuristic and of the optimal policy grows monotonically as the classifier quality decreases. We interpret this as the planning problem becoming *harder*, in the sense that computing an expensive policy (i.e, the optimal one) becomes necessary to limit the penalty in performance.

b) Lower estimation quality blurs the difference among the heuristics: While for good quality estimations the best heuristics ($\tau \approx 0.6$) are clearly preferable, for low-quality estimations half of the heuristics ($\tau > 0.5$, i.e., any relatively pessimistic policy) have the same performance.

c) The optimal policy performs better in mean and worst cases, but performs worse in the median case: The optimal policy is the best policy w.r.t. the average cost (unsurprisingly since this is how it is defined). Yet, in the median case, the best heuristics have a higher chance of being the best policy on random graphs; their larger chance to incur in very large costs penalizes their mean performance.

d) Policy performance heavily depends on the map: On random maps with 7 hidden edges (out of about 35), the average competitive ratio is small; in fact, for about 60% of the instances the cost is minimal for all policies. We may be tempted to conclude that it is not worth to develop high-quality traversability classifiers; yet on many maps, such as the two real maps we presented, the gaps are significantly larger (i.e., the problem is harder).

e) The choice of policy depends on the classifier quality and on the concrete implementation scenario: Concrete scenarios where a navigation algorithm is adopted carry secondary objectives and requirements, which guide the choice between an heuristic and the optimal policy. For high-quality estimations, when optimizing the performance in the median case, and when computational resources are constrained, heuristics are a very good choice. On the contrary, with low-quality estimations and when focusing on the average (or worst-case) performance, the optimal policy is the best choice.

VII. Conclusions

We presented a large-scale simulation campaign on millions of synthetically-generated maps and two real-world maps in the context of the Canadian Traveller Problem; our study evaluated the effects of two sources of uncertainty (inaccuracy of the classifier and number of hidden edges) on the cost of paths obtained by the optimal policy and a family of baseline reactive strategies.

We could observe that: (1) uncertainty makes the problem harder; (2) heuristics perform well with high-quality classifiers, even more so considering the median case; (3) when using heuristics, it is better to be slightly pessimistic ($\tau > 0.5$) than optimistic; (4) there is a large variability in relative policy performance on different graphs.

One straightforward extension of this work is to increase the size of the graphs, or the ratio of hidden edges, to verify, as we believe, that our analysis remain valid.[2] We would also like to investigate the impact of the graphs themselves on the policy performance, for instance of their density. Which graphs give rise to the most challenging planning instances? Are the distributions we used to generate graph samples representative of real-world maps?

Another very interesting direction to investigate is the role of calibration as real-world classifiers cannot be perfectly calibrated, e.g., to consider the impact of accurate but less trustworthy estimators.

[2]The optimal policy can only be computed for some tens of edges; sophisticated (but also computationally expensive) heuristics that perform a partial exploration of the search space [13] can be used to approximate the optimal policy on larger graphs.

REFERENCES

[1] W. Wang, M. Shen, J. Xu, W. Zhou, and J. Liu, "Visual traversability analysis for micro planetary rover," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2009, pp. 907–912.

[2] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning ground traversability from simulations," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1695–1702, 2018.

[3] K. Deb, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.

[4] Y. Zhang, D.-w. Gong, and J.-h. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomputing*, vol. 103, pp. 172–185, 2013.

[5] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1994, pp. 3310–3317.

[6] A. Bar-Noy and B. Schieber, "The Canadian Traveller Problem," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 1991, pp. 261–270.

[7] D. Fried, S. E. Shimony, A. Benbassat, and C. Wenner, "Complexity of Canadian traveler problem variants," *Theoretical Computer Science*, vol. 487, pp. 1–16, 2013.

[8] E. Nikolova and D. R. Karger, "Route Planning under Uncertainty - The Canadian Traveller Problem," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 969–974, 2008.

[9] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[10] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2013, pp. 1772–1780.

[11] P. Eyerich, T. Keller, and M. Helmert, "High-Quality Policies for the Canadian Traveler's Problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010, pp. 51–58.

[12] O. F. Sahin and V. Aksakalli, "A Comparison of Penalty and Rollout-Based Algorithms for the Canadian Traveler Problem," *International Journal of Machine Learning and Computing*, vol. 5, no. 4, p. 319, 2015.

[13] D. Ferguson, A. Stentz, and S. Thrun, "PAO* for planning with hidden state," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004, pp. 2840–2847.

[14] Z. Bnaya, A. Felner, and S. E. Shimony, "Canadian Traveler Problem with Remote Sensing," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009, pp. 437–442.

[15] Z. Bnaya, A. Felner, D. Fried, O. Maksin, and S. E. Shimony, "Repeated-task canadian traveler problem," *AI Communications*, vol. 28, no. 3, pp. 453–477, 2015.

[16] D. Dey, A. Kolobov, R. Caruana, E. Kamar, E. Horvitz, and A. Kapoor, "Gauss meets Canadian traveler: shortest-path problems with correlated natural dynamics," in *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2014, pp. 1101–1108.

[17] M. Kull, T. S. Filho, and P. Flach, "Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 623–631.

[18] W. J. Park and R. M. Kil, "Pattern Classification With Class Probability Output Network," *IEEE Transaction on Neural Networks*, vol. 20, no. 10, pp. 1659–1673, 2009.

[19] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2010, pp. 3121–3124.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[21] M. Wermelinger, P. Fankhauser, R. Diethelm, P. A. Krüsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1184–1189.