Shortest Path under Uncertainty: Exploration versus Exploitation

Zhan Wei Lim
Movel AI
zhanwei@movel.ai

David Hsu
Department of Computer Science,
National University of Singapore
dyhsu@comp.nus.edu.sg

Wee Sun Lee
Department of Computer Science,
National University of Singapore
leews@comp.nus.edu.sg

Abstract

In the Canadian Traveler Problem (CTP), a traveler seeks a shortest path to a destination through a road network, but unknown to the traveler, some roads may be blocked. This paper studies the Bayesian CTP (BCTP), in which road states are correlated with known prior probabilities and the traveler can infer the states of an unseen road from past observations of other correlated roads. As generalized shortest-path problems, CTP and BCTP have important practical applications. We show that BCTP is NP-complete and give a polynomial-time approximation algorithm, *Hedged Shortest Path under Determinization* (HSPD), which approximates an optimal solution with a polylogarithmic factor. Preliminary experiments show promising results. HSPD outperforms a widely used greedy algorithm and a state-of-the-art UCT-based search algorithm for CTP, especially when significant exploration is required.

1 INTRODUCTION

The Canadian Traveler Problem (CTP) is a generalization of the Shortest Path Problem. A traveler seeks a shortest path through a road network, modeled as a graph, from a source to a destination node. Some roads, however, are impassable. The traveler finds out the state of a road only upon reaching it. CTP has diverse applications in transportation planning, network routing, robot navigation, *etc.* [1, 2, 9, 18], but it is PSPACE-complete [19].

We introduce the Bayesian Canadian Traveler Problem (BCTP), a Bayesian variant of CTP, in which each configuration of the road network has known prior probability. A distinctive feature of BCTP is that road states may be correlated. For example, rain or snow may block all roads in a region; scheduled repair may block roads according to a known pattern. The traveler can infer the state of an unseen road from past observations of other correlated roads. BCTP is thus a more realistic model than CTP for shortest path finding under uncertainty in many applications.

To find a shortest path under uncertain road conditions, the traveler may seek to gain information on road conditions, or proceed directly towards the destination on the presumed best path, despite the uncertainty. The two objectives are sometimes in conflict, and this reflects the exploration-exploitation trade-off, a fundamental issue in reinforcement learning and planning under uncertainty. An optimal solution must balance exploration and exploitation carefully. BCTP is in fact a special case of model-based Bayesian reinforcement learning [5, 7], with deterministic transition dynamics, and provides a simplified setting for studying the exploration-exploitation trade-off.

We show that BCTP is NP-complete and give a polynomial-time approximation algorithm, *Hedged Shortest Path under Determinization* (HSPD), which approximates an optimal solution with a polylogarithmic factor.

HSPD uses two key ideas. The first is to determinize the road network graph according to the *most likely edge* (MLE) assumption: each edge is assumed passable if the marginal probability of it being passable is greater than 0.5. Determinization simplifies the problem by removing uncertainty. The MLE determinization has an important property: if the traveler follows a path in the determinized graph and encounters a road state incompatible with the MLE assumption, then the probability of compatible road configurations is reduced by at least a half. Consequently, the MLE assumption is wrong for at most a logarithmic number of times. After that, the true states of all roads are known, leaving us a simple shortest path problem.

Let us call the shortest path in a determinized graph the *exploitation path*. The exploitation path may be far from being optimal, as all edges with marginal probability less than 0.5 are removed. In fact, the destination node may not even be reachable in the determinized graph. Our second key idea handles this by hedging the exploitation path with the *exploration path*. The exploration path in a determinized graph is the shortest path that gathers information and reduces the probability of incompatible road configurations by at least a half. Somewhat surprisingly, the shorter of the exploitation path and the exploration path has length within a constant factor of the expected length of an adaptive optimal solution.

HSPD is recursive and computes, in each step, the exploitation and exploration paths. It traverses the shorter of the two paths, until it encounters a road state incompatible with the MLE determinization assumption. HSPD then updates the probability distribution over the road configurations, taking advantage of any correlation present, and repeats until it reaches the goal.

We compare HSPD with two alternatives. One is a widely used greedy algorithm that makes an optimistic assumption on road conditions, and the other is a UCT-based heuristic search algorithm, which has achieved some of the best empirical results [8]. The empirical comparison shows promising results for HSPD. It outperforms the greedy algorithm, as expected. It also outperforms the UCT algorithms, even when the UCT algorithm is given substantially more computation time. The performance gap is particularly large, when significant exploration is required.

2 RELATED WORK

The BCTP is tied closely to many planning problems in partially observable domains. One such problem is the *Active Learning* [23] problem where a learner adaptively queries the labels of unlabeled training data to learn a classifier while using as few queries as possible. The unlabeled training data is analogous to the roads in BCTP. However, active learning is only concerned with gathering information efficiently and does not involve proceeding to a goal state whereas BCTP has to balance gathering information and proceeding to the goal. Furthermore, in active learning, taking an action does not affect the future cost of other actions, whereas in BCTP, going to a different location affects the cost of visiting other sites from that location. This latter property is present in adaptive Informative Path Planning (IPP) [15] and the adaptive Travelling Salesperson Problem (TSP) [11]. Both adaptive IPP and adaptive TSP extend active learning to path planning problems where actions move the agent to the next locations in a metric space in order to collect observations and identify the target hypothesis with minimum expected cost. Like active learning, adaptive IPP and adaptive TSP also do not have a goal other than gathering information, hence do not have to balance exploration and exploitation.

Another closely related problem is the Bayesian reinforcement learning problem [5, 7]. In the Bayesian reinforcement learning problem, the environment is modeled as an unknown Markov decision process (MDP) but we have a prior probability distribution over the MDP models. An agent that maximizes its expected long-term reward has to balance exploration and exploitation. As in BCTP, an agent may lose potential higher reward if it explores more than necessary to learn about the true model. On the other hand, it may also lose higher future reward if it stops learning about the model too early and exploits its current knowledge of the model for rewards. The Bayesian reinforcement learning is more complex as its uncertainty is over MDP models, a much richer model than the graphs in BCTP. The BCTP may serve as a simpler problem to further our understanding of Bayesian reinforcement learning problem.

A number of variants of Canadian Traveler problem have been proposed over the year. In [18], an efficient exact algorithm was given for the special case where the graphs contain disjoint paths from source to destination with random two-valued edge costs. Another variant is the remote-sensing Canadian Traveler Problem [3] where an agent can pay a sensing cost to reveal whether a road not incident on its current site is passable. This problem appear to be harder than the Canadian Traveler problem; it is NP-hard even in the special case of disjoint path [10]. BCTP is similar to the remote-sensing Canadian Traveler problem in the sense that the agent can learn about states of roads not incident on its current site. However, BCTP achieves that by correlation between states of road instead of directly sensing it. The most similar variant to BCTP is the one proposed in [21] where the edge costs are correlated. In fact, BCTP can be expressed as a special case of the model in [21] by setting the edge cost to infinity to make an edge impassable. In [21], an approximation algorithm that achieves an approximation factor of $\min(N, R)$, where N is number of graph nodes and R is the number of realizations of edge costs, is described. Our algorithm achieve a polylogarithmic approximation factor (see Theorem 2), which is better in many situations. In a recent variant, [6] modeled the edge costs using Gaussian Process. The Canadian Hacker Problem is a variant of the Canadian Traveler Problem applied to cybersecurity, where the aim is to reach a particular host by compromising network nodes [12].

Our algorithm and analysis follows the intuition of assuming the most likely observation developed in [16] and [15]. However, in those works the graph connectivity is assumed to be known while in BCTP, we assume unknown graph connectivity. Maximum likelihood determinization has also been used in Markov decision process (MDP) planning [24] and partially observable Markov decision process (POMDP) planning [20]. Those cases are planning problems where the probabilistic models are assumed to be known, unlike the Canadian Traveler Problem where the graph needs to be learned while the agent is trying to reach the goal.

Both Bayesian and the original Canadian Traveler Problem can be formulated as partially observable Markov decision process (POMDP) [2] models. While modern POMDP solvers are able to scale up to large state spaces, solving a POMDP remains difficult in practice.

3 BAYESIAN CANADIAN TRAVELER PROBLEM

We represent a road map for a Bayesian Canadian Traveler Problem (BCTP) as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ where $\mathcal{V}$ (the set of nodes) is the set of sites, $\mathcal{E}$ (the set of edges) is the set of roads in the road map, and $\mathcal{W} : \mathcal{E} \to \boldsymbol{R}$ gives the length of the roads on the road map (weight of each edge). While we motivate the problem on road maps, we only require that the weight function forms a metric.

For each edge $e \in \mathcal{E}$, let X_e be the random variable that equals its state. *i.e.*, $X_e = 1$ if the road corresponding to edge e is passable and $X_e = 0$ otherwise. We denote the vector of the states of a set of edges $\mathcal{A}$ as $x_{\mathcal{A}}$. In particular $x_{\mathcal{E}}$ denotes the configuration of edges in the entire graph. We assume that there are n possible configurations $x_{\mathcal{E}}$ for the entire graphs and call the set of possible configurations the hypothesis space $\mathcal{H} = \{x^1_{\mathcal{E}}, \ldots, x^n_{\mathcal{E}}\}$, represented as a n by m binary matrix, where m is the number of edges. In addition, the prior probability $p(h = x_{\mathcal{E}})$ for each hypothesis is given in a vector p. Overall, the Bayesian Canadian Traveler Problem is specified by the tuple $(\mathcal{G}, \mathcal{H}, p, s, t)$ where $\mathcal{G}$ is the graph, $\mathcal{H}$ is the hypothesis space, p is the prior probability, s is the starting node, and t is the destination node. We assume that the destination is reachable for all configurations.

When a traveler arrives at node v, he observes the states of incident edges at node v. For *e.g.*, if edges e_1, e_2 are incident on v, then the traveler observes a vector of states $\langle X_{e_1}, X_{e_2} \rangle$. To simplify notations, we refer to the vector of edge states observed at a site v as an observation $x_v = x_{\mathcal{N}(v)}$, where $\mathcal{N}(v) \subseteq \mathcal{E}$ is the set of edges incident on node v.

The solution to BCTP can be represented as a policy tree π, where the nodes in the policy tree specify a site $v \in \mathcal{V}$ to visit. Each branch is labeled by an observation o_v. To follow a policy specified by a policy tree π, a traveler first travels to the site specified by the current policy node of π, v_i. The traveler then observes the states of edges incident on v_i and takes the branch matching his observation o_{v_i} to go to the next policy node v_j. The procedure is repeated on the subtree rooted at v_j until the traveler reaches a terminal node in policy tree π, which is always labeled with the destination node t. Any two nodes connected in policy tree π must also be connected in G for the policy to be valid. The root node of a policy π must be labeled s; we assume that the traveler is initially located at the graph node s and starts following the policy at the root of the policy tree by observing the states of edges incident to s.

The cost incurred by a traveler is defined as $C(\tau) = \sum_{e \in \tau} \mathcal{W}(e)$, where $\tau = e_1, e_2, \ldots$ is the sequence of graph edges traversed by the traveler.

BCTP seeks an adaptive path to reach destination node t from starting node s with the least expected cost. The expected cost is defined as:

$$\mathrm{E}_H[C(\pi(h))] = \sum_{h \in \mathcal{H}} p(h) C(\pi(h)),$$

where $\pi(h)$ is the path in policy tree taken when $H = h$.

3.1 Computational Complexity

The decision version of BCTP problem is NP-complete. We prove this by a reduction from the Optimal Decision Tree (ODT) problem.

In the ODT problem, $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ is a finite set of objects and $\mathcal{E} = \{e_1, \ldots, e_m\}$ is a finite set of tests. A test e_i reveals an outcome $x_i \in \{0, 1\}$ that depends on the unknown object $h \in \mathcal{H}$. We let function $p(e_i, h)$ map a test e_i and true object h to its outcome. In the ODT problem, we aim to find a policy to identify an element in $\mathcal{H}$ with the least expected number of tests, when the hypotheses are uniformly distributed. The policy is a binary decision tree where the root and internal nodes specify a test and the terminal nodes give the true object $h \in \mathcal{H}$. The decision version of the problem, which asks whether there is a policy with expected cost of less than or equal to w is NP-complete [13].

Theorem 1. *We define the decision version of Bayesian Canadian Traveler Problem as the question of whether there is a policy with expected cost less than or equal w. The decision version of BCTP is NP-complete.*

Proof. The solution of a BCTP problem can be represented as a policy tree. To see that an optimal policy tree is of polynomial size, note that every hypothesis can be associated with a path to a leaf in the policy tree, and the size of the tree is no more that the total size of all these paths. Furthermore, the number of observations that can cause branching on a path to a leaf is at most the number of vertices in $\mathcal{G}$ and the path between two such observation locations does not contain a cycle in the graph.

Finally, we can compute the expected cost of an optimal policy in polynomial time by doing a weighted sum of the cost of each $h \in \mathcal{H}$, showing that the problem is in NP.

We prove that BCTP is NP-hard by showing that ODT is polynomial time reducible to BCTP.

Given an instance of ODT$(\mathcal{H}, \mathcal{E})$, we construct an instance of BCTP $(\mathcal{G} = (\mathcal{V}, \mathcal{E}'), \mathcal{H}', P, s, t,)$ as follows.

The graph G consists of two clusters of edges as shown in Figure 1. Given each $h_i \in \mathcal{H}$, we construct the corresponding $h'_i \in \mathcal{H}'$ as follows. The first cluster of edges $e_1, \ldots, e_m, e_{m+1}, \ldots e_{2m}$ mirrors the tests in ODT. Edges $e_1, \ldots, e_m$ are always passable. An agent after traversing an edge $e_j \in \{e_1, \ldots, e_m\}$ will observe whether e_{m+j} is passable. The edge e_{m+j} will be passable if and only if the outcome of test e_j under h_i is 1. The second cluster of edges consists of n disjoint paths from v_i to destination v_t. The first edge in the i-path e_{2m+n+i} is always passable. The second edge e_{2m+i} will be passable if and only if the unknown object is h_i. Hence, only one out of the n paths is passable. All edges has cost of 1 except $e_{2m+n+1}, \ldots, e_{2m+2n}$ where they have cost $2m$. Finally, all $h' \in \mathcal{H}'$ are given equal probability.

We now argue that the expected cost of the ODT instance is less than or equal to w if and only if the expected cost of the BCTP instance is less than or equal to $2m+2+2w$.

First, we note that if the expected cost of the ODT instance is no more than w, then we can traverse the left cluster using the policy tree of the ODT instance and identify the correct h' with cost no more than $2w$, then go directly to t with an additional cost of $2m+2$. We now argue the converse that if the expected cost of the BCTP instance is no more than $2m+2+2w$, the expected of the ODT instance is no more than w. The case $w \geq m$ is vacuous as the ODT solution can always be found with cost no more than m by running each of the m tests. If $w < m$ then by our assumption, the expected cost of the BCTP instance is less than $2m+2+2m$. This can only be done by identifying the hypothesis first before traversing to the target. To see this, we note that as long as the hypothesis has not been identified, there are at least two equally likely path to reach the target, and the expected cost of any policy that tries to go to the target directly when there are at least two equally likely path is at least $4m+2$. Hence, the policy must identify h' first at a cost no more than $2w$ before traversing to the target with an additional cost of $2m+2$.

Thus ODT is reducible to BCTP in polynomial time, and BCTP is NP-complete. □

We note that it is possible to encode $\mathcal{H}$ using representations other than a n by m binary matrix. In particular, factored representations may result in more compact representations of the distribution of hypotheses. With such representations, the BCTP may no longer be in NP. In particular, the original Canadian Traveler problem is PSPACE-complete [10]. Also, in general, there may be no compact representation of the posterior distribution even when there is a compact representation of the prior distribution, unless the distribution has some special structure.

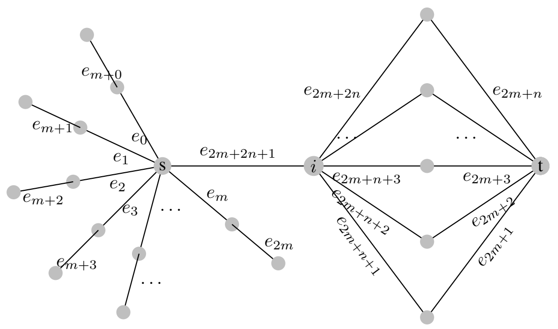

Figure 1: Reduction from Optimal Decision Tree problem

4 HEDGED SHORTEST PATH UNDER DETERMINIZATION

HSPD is a recursive algorithm. In each recursive step, we determinize the state of every edge using the MLE assumption to obtain a determinized graph $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}}, \mathcal{W})$. This graph is the same as the graph G of BCTP except that its edges whose probability of being passable is less than a half is removed. More formally, $\hat{\mathcal{E}} = \{e \in \mathcal{E} | P(X_e = 1) \geq 0.5\}$.

The idea of determinization is important. First, determinization gives a computational simpler problem whose solution is a path, as opposed to a policy tree that needs to account for all observations at every step. Second, if the observation received disagrees with the determinized graph while following the chosen path, then at least half of the probability mass of the current hypotheses would be eliminated. Furthermore, this happens at most a logarithmic number of times before we identify the true road network.

We call the solution of the shortest path problem in the determinized graph the exploitation path. Unfortunately, the exploitation path alone is not enough to solve the problem. In fact, a path to the goal may not exist in the determinized graph, even if a path exists under the true hypothesis. Furthermore, the exploitation path may be longer than necessary as some passable roads are assumed un-passable.

To hedge against the possibility of a poor exploitation path, HSPD also plans an exploration path in the determinized graph. To obtain the exploration path, we solve the problem of finding the quickest way to gather information and reduce the probability mass of consistent hypotheses by half. Let the version space size function be $\varphi(x_{\mathcal{A}}) = \sum_{h \in \mathcal{H} \text{ consistent with } x_{\mathcal{A}}} p(H = h)$. Let $\hat{\mathcal{G}}_{\mathcal{A}} = \langle \mathbf{1}(e \in \hat{\mathcal{E}}) \rangle_{e \in \mathcal{A}}$ be the vector of edge states of the set of edges $\mathcal{A}$. For convenience, we normalize the probabilities to get conditional probabilities such that $p'(h) = P(H = h|x_{\mathcal{A}}) = \frac{P(H=h)}{\varphi(x_{\mathcal{A}})}$, and use p' as the input to each recursive step. The information gathering problem seeks to find the shortest path through $\hat{\mathcal{G}}$ such that $\varphi(\hat{\mathcal{G}}_{\mathcal{A}}) \leq 0.5$, where $\mathcal{A}$ is the set of edges observed along the path. The version space size function φ is a submodular function and finding the shortest path to reduce version space size is a submodular orienteering problem.

In a submodular orienteering problem, there is a set of locations X, a metric d that gives the distance between any pair of locations $x, x' \in X$, a starting location r, and a submodular function f of the set of locations. The goal of submodular orienteering problem is to find a tour starting from r that covers the function f. To solve for the exploration path, we give determinized graph $\hat{\mathcal{G}}$, weight of the edges $\mathcal{W}$, version space function φ, and starting node s as input to the submodular orienteering problem. Solving the submodular orienteering problem exactly is computationally hard. We use the polynomial-time approximation procedure described in [17].

Let τ^* be the exploitation path and τ_φ be the exploration path in $\hat{\mathcal{G}}$. HSPD chooses the shorter path τ out of the two paths τ_φ and τ^* or chooses τ_φ if there is no path from s to t on $\hat{\mathcal{G}}$. HSPD then traverses τ until the end unless we receive an observation that violates the MLE assumption, *i.e.*, when we receive an observation o_v at site v such that $o_v \neq \hat{\mathcal{G}}_{\mathcal{N}(v)}$; along the way we add any received observation to the observed set $\mathcal{A}$, denoting the operation as $x_{\mathcal{A}} \| o_v$.

If we had traversed the entire τ^* then we would have reached the destination node t and we are done. Otherwise, we go on to the next recursive step that works on the posterior distribution p' given the states of edges observed $x_{\mathcal{A}}$.

The pseudo-code of the algorithm is shown as Algorithm 1. The algorithm, as presented, always return to the starting site after each recursive call. This simplifies analysis. A practical version may skip this step and resume from the same site after each recursive call.

To fine tune the balance between exploration and exploitation we introduce a parameter $\alpha \geq 1$ that scales the length of the exploration tour τ_φ at Line 7 of Algorithm 1 such that $\tau \leftarrow \arg\min(W(\tau^*), \alpha W(\tau_\varphi))$. The bigger the value of α, the more HSPD will favor exploitation over exploration.

Algorithm 1 HSPD

```
procedure RECURSEHSPD($p$)
    construct $\hat{\mathcal{G}}$ using most likely observation determinization
    $\tau^* \leftarrow$ SHORTESTPATH($\hat{\mathcal{G}}$)
    if $\max_{h \in \mathcal{H}} P(H = h) = 1.0$ then
        follow tour $\tau^*$
    $\tau_\varphi \leftarrow$ ORIENTEER($\hat{\mathcal{G}}, \mathcal{W}, \varphi, s$)
    $\tau \leftarrow \arg\min(W(\tau^*), W(\tau_\varphi))$
    $x_{\mathcal{A}} \leftarrow$ EXECUTEPLAN($\tau$)
    $p' = P(H|x_{\mathcal{A}})$
    RECURSEHSPD($p'$)
procedure EXECUTEPLAN($\tau$)
    repeat
        Visit next site $v$ in $\tau$ and observe $o_v$.
        $\mathcal{A} \leftarrow \mathcal{N}(v)$
        $x_{\mathcal{A}} \leftarrow x_{\mathcal{A}} \| o_v$.
    until $\hat{\mathcal{G}}_{\mathcal{N}(v)} \neq o_v$ or end of tour.
    Move to node $s$ by tracing back its step
    return $x_{\mathcal{A}}$
```

5 ANALYSIS

This section provides the performance guarantee for HSPD. Each recursive step divides the BCTP problem into smaller subproblems such that each subproblem consists of the subset of the hypotheses that are consistent with observations received, and its prior distribution is the posterior of the original problem given the observations. Note that at run time, only the subproblem corresponding the actual observations received needs to be solve. We show that each recursive step either reduces the size of the problem by a constant factor (as measured by probability mass of consistent hypotheses) or solves the problem completely (by reaching the destination). Furthermore, we show that the distance traveled in each recursive call can be bounded by a constant factor of the cost of the optimal policy. As each recursive call reduces the size of the subsequent subproblems by a constant factor, we can bound the number of the recursive calls needed to solve the BCTP problem and provide a performance bound.

We begin our analysis by bounding the distance traveled in each recursive call. We extract a subpath σ from the optimal solution π^* by following the most likely branch of the optimal policy tree and truncate just before the probability of traversing the path becomes less than 0.5. Lemma 1 gives the properties of this path. Lemma 2 shows that the subpath σ is present on $\hat{\mathcal{G}}$.

Lemma 1. *There is a subpath $\sigma = v_1, \ldots, v_k$ that is* *traversed with probability at least 0.5 under the optimal policy π^*.*

Furthermore, either (1) σ ends at destination node t or (2) $\varphi(x_{v_1}\|\ldots\|x_{v_{k-1}}\|x_{v_k}) \leq 0.5$, where $x_{v_1},\ldots,x_{v_{k-1}}$ is the sequence of observations that leads to the subpath σ under π^*, and x_{v_k} is any observation.

Proof. Let $p(\sigma|\pi^*)$ be the probability of traversing a subpath σ under optimal policy π^*. It is equal to the joint probability of the receiving the observation labels of branches in π^* that lead to the subpath σ, $p(\sigma|\pi^*) = p(x_{v_1}, x_{v_2}, \ldots, x_{v_{k-1}})$, where $x_{v_1}, x_{v_2}, \ldots, x_{v_{k-1}}$ are observation labels on the branches along subpath $\sigma = v_1, v_2, \ldots, v_k$.

We show that such a subpath σ exists by constructing it from the optimal policy π^*. Let σ' be the path from the root node to terminal node in optimal policy tree π^* by assuming we always receive the most likely observation $x_v^* = \arg\max_{x_v} p(x_v|x_{\mathcal{A}})$, where $\mathcal{A}$ is the set of edges observed so far. If $p(\sigma'|\pi^*) \geq 0.5$ then we simply let $\sigma = \sigma'$. Otherwise, we truncate the path σ' up to the node before the probability of traversing it becomes less than 0.5. Specifically, suppose $\sigma' = v_1, v_2, \ldots, v_k, v_{k+1}, \ldots$ we truncate σ' up to until a node v_k such that $p(v_1, v_2, \ldots, v_k, v_{k+1}|\pi^*) \leq 0.5$ and $p(v_1, v_2, \ldots, v_k|\pi^*) > 0.5$.

The subpath σ satisfies property (1) if σ ends at terminal node. Otherwise it satisfies property (2) since the probability of the path going one node further down the most likely branch than σ is $p(v_1, v_2, \ldots, v_k, v_{k+1}|\pi^*) \leq 0.5$ and the probability of any observation x_{v_k} is at most the probability of the most likely observation branch, *i.e.*, $p(x_{v_k}|x_{\mathcal{A}}) \leq p(x^*_{v_k}|x_{\mathcal{A}})$. □

Lemma 2. *The sequence of observations $x_{v_1},\ldots,x_{v_{k-1}}$ that leads to subpath σ under optimal policy π^* is consistent with the vector of edge states that $\hat{\mathcal{G}}$ generates, i.e., $x_{v_1}\|\ldots\|x_{v_{k-1}} = \hat{\mathcal{G}}_{\mathcal{A}}$, where $\mathcal{A}$ is the set of edges observed at every node in σ except the last one. As a corollary, $\hat{\mathcal{G}}$ contains all edges necessary to connect the subpath σ.*

Proof. We prove the main statement by contradiction. Suppose there is an edge e whose state is observed at a node $v \in \sigma$ and its state is not consistent with $\hat{\mathcal{G}}$, *i.e.*, $x_e \neq \mathbf{1}(e \in \hat{\mathcal{E}})$. By definition of $\hat{\mathcal{G}}$, $p(X_e \neq \mathbf{1}(e \in \hat{\mathcal{E}})) < 0.5$. However, lemma 1 states that $p(\sigma|\pi^*) \geq 0.5$ and $p(\sigma|\pi^*) = p(x_{v_1}, x_{v_2}, \ldots, x_{v_{k-1}}) < p(X_e \neq \mathbf{1}(e \in \hat{\mathcal{E}}))$. Thus, we reached a contradiction and the observation sequence $x_{v_1},\ldots,x_{v_{k-1}}$ is consistent with $\hat{\mathcal{G}}$.

The corollary is true because any edge e needed to connect the subpath σ must be observed as $x_e = 1$ in the observation sequence $x_{v_1},\ldots,x_{v_{k-1}}$ and $\hat{\mathcal{G}}$ is consistent with the observation sequence. □

Since the subpath σ is traversed with probability at least 0.5 under the optimal policy and is a feasible solution to the tour required by the recursive call, we can bound the length of the tour generated in each recursive call by a factor of the expected cost of the optimal policy.

Lemma 3. *Assume we compute the optimal solution to τ^* and τ_φ in each recursive call. The distance traveled for each recursive call is at most $4C(\pi^*)$.*

Proof. From lemma 1 and lemma 2, if σ terminates at the destination t, then it is a feasible solution to the shortest path problem on graph $\hat{\mathcal{G}}$. Otherwise, we show that $\varphi(\hat{\mathcal{G}}_{\mathcal{A}}) \leq 0.5$ and hence σ is a feasible solution to the version space reduction problem, where $\mathcal{A}$ is the set of edges observed in σ.

Lemma 2 states that the observations received on σ under π^* are consistent with $\hat{\mathcal{G}}$ up to the second last node in σ. We get $\hat{\mathcal{G}}_{\mathcal{A}} = x_{v_1}\|\ldots\|x_{v_{k-1}}\|x'_{v_k}$ where $x'_{v_k} = \hat{\mathcal{G}}_{\mathcal{N}(v_k)}$. We need to prove two cases: when the observation generated by $\hat{\mathcal{G}}$ at v_k is consistent with some outgoing branch from v_k and when it is not. In the first case, $x'_{v_k} = x_{v_k}$, where x_{v_k} is some outgoing branch on π^* from v_k. If σ does not terminate at t, then from lemma 1, $\varphi(x_{v_1}\|\ldots\|x_{v_{k-1}}\|x_{v_k}) \leq 0.5$. If, on the other hand, x'_{v_k} is not consistent with any outgoing branch, the version space is immediately set to zero. Hence σ is a feasible solution to the version space reduction problem.

The cost incurred in each recursive call is $W = \min(C(\tau^*, \tau_\varphi)$. If σ terminates at the destination t, then $W \leq C(\tau^*) \leq C(\sigma)$. Otherwise, $W \leq C(\tau_\varphi) \leq C(\sigma)$. Furthermore, subpath σ is traversed with probability at least 0.5 in the optimal policy π^*, then

$$C(\pi^*) \geq 0.5C(\sigma) \geq 0.5W$$
$$W \leq 2C(\pi^*)$$

At the end of ExecutePlan, if it did not reach the destination t, the agent traces its path back to node s for the recursive call. Hence, the distance traveled for each recursive call is at most $4C(\pi^*)$. □

Lemma 4. *Let $x_{\mathcal{A}}$ be the states observed after a recursive call of HSPD. Then either: (1) recursive call reached the destination node t (2) or we have reduced the probability mass of consistent hypotheses by at least half such that $\varphi(x_{\mathcal{A}}) \leq 0.5$.*

Proof. Suppose the agent completes the tour τ, if the tour chosen in the recursive call is τ^* then it would reach the destination node since τ^* ends at destination node. Otherwise, the tour chosen is τ_φ. As the tour did not terminate early, all states observed must be consistent with the observations of the determinized graph $\hat{\mathcal{G}}$, $x_{\mathcal{A}} = \hat{\mathcal{G}}_{\mathcal{A}}$. By the problem defintion of τ_φ, we have $\varphi(\hat{\mathcal{G}}_{\mathcal{A}}) \leq 0.5$.

Hence, we have reduced the probability mass of consistent hypotheses by at least half.

For the case when the agent did not complete the tour τ, then it must encounter an edge e' whose state $x_{e'}$ is inconsistent with its the determinized graph $\hat{\mathcal{G}}$. From the definition of the set of edges in the determinized graph, $\hat{\mathcal{E}} = \{e \in \mathcal{E} | P(X_e = 1) \geq 0.5\}$, we get the probability of the observed edge state $x_{e'}$ is $P(X_{e'} = x_{e'}) < 0.5$. Since $\varphi(x_{\mathcal{A}}) \leq \sum_{h \in \mathcal{H}} P(H = h | x_{e'}) = P(X_{e'} = x_{e'}) < 0.5$, we have reduce the probability mass of consistent hypotheses by at least half as well. $\square$

After dividing the BCTP problem in smaller subproblems, we show that the sum of cost of optimally solving each subproblem weighted by the probability of its occurence is at most the expected cost of the optimal policy of the original problem. In short, the problem does not become harder when it becomes smaller.

Lemma 5. *Let π^* be an optimal policy for an instance of BCTP $(\mathcal{G}, \mathcal{H}, p, s, t)$. Let $\{\mathcal{H}_1, \dots, \mathcal{H}_K\}$ be a partition of the hypotheses $\mathcal{H}$. We denote the sum of probabilities of a subset $\mathcal{H}_i$ as $p(\mathcal{H}_i) = \sum_{h' \in \mathcal{H}_i} p(H = h')$. For each $\mathcal{H}_i$ we define the i-th subproblem, BCTP $(\mathcal{G}, \mathcal{H}_i, p_i, s, t)$ where*

$$p_i = p(H) \triangleq \begin{cases} p(H = h)/p(\mathcal{H}_i) & \textit{if } h \in \mathcal{H}_i \\ 0 & \textit{otherwise} \end{cases}.$$

Let π_i^* be the optimal policy for the ith subproblem. Then we have $\sum_{i=1}^{K} p(\mathcal{H}_i) C(\pi_i^*) \leq C(\pi^*)$.

Proof. We can extract a feasible solution π_i for every subproblem i from the optimal policy of the original problem π^* by combining all paths taken under π^* for each hypothesis $h' \in \mathcal{H}_i$. The combination of these paths is a subtree of π^*. Hence,

$$\begin{aligned} \sum_{i=1}^{K} p(\mathcal{H}_i) C(\pi_i^*) &\leq \sum_{i=1}^{K} p(\mathcal{H}_i) C(\pi_i) \\ &\leq \sum_{i=1}^{K} p(\mathcal{H}_i) \sum_{h \in \mathcal{H}_i} \frac{p(h)}{p(\mathcal{H}_i)} \cdot C(\pi_i, h) \\ &= \sum_{h \in \mathcal{H}} p(h) C(\pi^*, h) = C(\pi^*). \end{aligned}$$

$\square$

Because each recursive call divides the problem into sufficiently small ones by halving the probability mass of consistent hypotheses except when it reaches the destination, we can bound the number of recursive calls required.

Lemma 6. *Suppose we solve the submodular orienteering problem in HSPD optimally. For an instance of $BCTP(G, \mathcal{H}, p, s, t)$, HSPD computes a policy π such that*

$$C(\pi) \leq 4(\log \delta + 1) C(\pi^*),$$

where $\delta = 1/p_{min}$ and $p_{min} = \min_{h \in \mathcal{H}} p(h)$ is the prior probability of the least likely edge configuration.

Proof. From lemma 4, every recursive call either reaches the destination or reduces the probability mass of consistent hypotheses by at least half. In the worst case, the recursive call is repeated until there is only one consistent hypothesis left and then it proceeds to the destination using shortest path on $\hat{\mathcal{G}}$. Hence, the number of recursive call is at most $\log \delta + 1$.

We prove the bound on expected cost of policy π by induction on the number of recursives call. In the base case $i = 1$, lemma 3 gives us $C(\pi) \leq 4C(\pi^*)$. Suppose that $C(\pi) \leq 4(i-1)C(\pi^*)$ when there are at most $i - 1$ recursive calls. We now consider the case where there are i recursive call. The first recursive call partitions $\mathcal{H}$ into K mutually exclusive subsets $\mathcal{H}_1, \dots, \mathcal{H}_K$ by the location in the tour when it is terminated. Each subset $\mathcal{H}_k$ induce a subproblem k of the original BCTP and it takes at most $i - 1$ recursive calls for each subproblem. By lemma 3, the first call incurs a cost at most $4C(\pi^*)$. By the induction hypothesis, each subproblem k incurs a cost at most $4(i-1)C(\pi_k^*)$. With lemma 5, the total cost is at most $4iC(\pi^*)$. $\square$

Finally, we substitute the optimal orienteering algorithm with a polynomial time approximation and add in the tuning factor α to get the final approximation bound.

Theorem 2. *For an instance of BCTP $(\mathcal{G}, \mathcal{H}, p, s, t)$, assume that prior probability distribution p is represented as non-negative integers with $\sum_{\mathcal{H}} p(h) = P$. HSPD with scaling parameter $\alpha \geq 1$ computes a policy π in polynomial time such that*

$$C(\pi) \in O(\alpha \log |\mathcal{V}|^{2+\epsilon} \log P (\log P + 1)) C(\pi^*)$$

Proof. In each recursive call, HSPD uses a polynomial time β-approximation algorithm in each recursive call to solve the version space reduction orienteering problem to get τ_φ. Let W be the length of the tour chosen in a recursive call when we use an approximation algorithm and let W_φ^* be the length of the optimal version space reduction tour. Then $W = \min(\beta W_\varphi^*, W(\tau^*)) \leq \beta \min(W_\varphi^*, W(\tau^*)) \leq 4\beta C(\pi^*)$. Using lemma 6, we get the approximation bound for HSPD to be $4\beta \log(P + 1) C(\pi^*)$.

The polynomial time approximation for the version space reduction orienteering problem is based on the greedy approximation algorithm in [4] (see [17] for details). The approximation factor for the greedy algorithm is $O(\log |\mathcal{V}|^{2+\epsilon} \log P)$. The scaling factor α can be seen as an additional factor to approximating τ_φ since it makes the tour looks longer. Putting all the factors together gives us the final approximation bound. □

The performance bound of HSPD is dependent on the probability of the least likely edge configuration. It is useful when the number of configurations is moderate but may be vacuous if the outcomes of the edges are independent of each other as in the case of the original CTP. In the experiment section, we show that HSPD can still give good performance with appropriate value of α on a mixture model with a very large number of configurations. The bound show that a bigger value of α gives a worse performance guarantee. However, in practice the parameter α helps to fine tune the balance between exploration and exploitation to give better empirical performance.

6 EXPERIMENTS

6.1 Setup

We implement two baseline algorithms, Optimistic and Upper Confidence Tree (UCT) [14], to compare the performance of HSPD. Optimistic plans a path by assuming a optimistic graph where all roads are passable unless it can be inferred to be otherwise from the observations. The algorithm takes the first step of the path, receives observation and re-plans on an updated optimistic graph where roads observed to be impassable are removed. UCT is a general Monte-Carlo tree search algorithm that uses the upper confidence bound to guide the search. It has been successfully applied to many MDP and POMDP problems [22]. We implement the variant of the UCT algorithm in [8] that is designed for good performance on the Canadian Traveler Problem. This variant of UCT uses Optimistic as rollout policy and incorporates two modifications to place more emphasis on the heuristic value of Optimistic in its search.

It may be tempting to implement a baseline that simply uses shortest path on graph $\hat{\mathcal{G}}$ of the deterministic instance of BCTP we defined in HSPD. That is, HSPD without the version space reduction part. However, this is an incomplete solution as the graph $\hat{\mathcal{G}}$ may not necessary contains a path from the agent's current site to the destination.

In the first experiment, we implement a BCTP that is a reduction from a optimal decision tree problem using Theorem 1. The original optimal decision tree problem has 10 tests and 25 hypotheses. The outcomes of the tests for each hypothesis are randomly generated at the beginning.

The second experiment simulates a road network affected by snow. The road network in the experiment is a 10 by 10 grid and every road has length 2. In each simulation, the ground truth road blockages are generated using a mixture model with 100 components. Each component is represented by a generating template, a configuration of blockages that is most likely for that component. To generate the ground truth road network in each simulation, we first sample one of the 100 templates with equal probability. Every road has a 0.9 chance of taking on the state as determined by the sampled template. If the state of a road is not taken from the template, the road state is randomly sampled with a 0.9 chance of being passable and a 0.1 chance of being blocked. To create a template, we simulate snowfall and its effect on the road by randomly choosing an affected area and blocking each road in the affected area with probability 0.5. The affected area is chosen by picking a rectangle whose side lengths are randomly selected to be between 3 and 10, and then randomly placing the rectangle on the 10 by 10 grid. To ensure the BCTP is sufficiently hard, for every template, we repeatedly simulate a snowfall (block each edge with probability 0.5) until more than 30 percent of the roads are blocked. As a finite mixture model, the posterior distribution of this model can be maintained in closed form.

We run HSPD on three different values of α: 1.0, 4.0, and 8.0. For the UCT algorithm, we use 100 and 300 rollouts per step and use an exploration constant of 5.0. For the first experiment, we run every hypothesis in the ODT problem once for Optimistic and HSPD. We run each hypothesis in the ODT 20 times for UCT as it is a randomized algorithm. For the second experiment, we generate 100 scenarios to get the correlated component of the mixture model and then sample 500 blockage configurations from the mixture model for evaluation and we run one simulation for each sample.

The results for the average path cost are shown in Table 1. We also report the 95% confidence and the average time taken per run in seconds. We limit the number of steps per run to 140 for the first experiment and 100 for the second experiment, and count a run as failure if the algorithm does not reach the destination within these number of steps. Failed runs are excluded in the calculation average path cost.

All algorithms are implemented in Clojure language. While care has been taken for efficiency, they are not highly optimized for performance.

6.2 Results

Our algorithm, HSPD, shows promising results compared to the other algorithms across both experiments. In the Optimal Decision Tree reduction experiment (Figure 1), out of the n paths that leads to the goal, only

Table 1: Average Cost

	HSPD $\alpha = 1.0$	HSPD $\alpha = 4.0$	HSPD $\alpha = 8.0$	Optimistic	UCT (100)	UCT (300)
ODT Reduction						
Cost	31.7 ± 0.64	31.7 ± 0.64	31.5 ± 0.456	502 ± 58.9	$566.\pm 32.1$	579 ± 29.1
Time (sec)	6.8	8.8	8.8	0.002	352	1069
Road Network						
Cost	63.3 ± 1.42	44.0 ± 0.774	38.9 ± 0.642	59.1 ± 1.59	44.6 ± 1.55	43.5 ± 1.58
Time (sec)	85.3	71.6	69.1	0.04	85.4	216

the one that corresponds to the true hypothesis is passable. The optimal strategy requires exploration using the "test" edges (e_0 to e_m) on the left to determine the true hypothesis before taking the path corresponding to the identified hypothesis. HSPD has the best result across different values of α. Optimistic performs badly for this problem because it does not gather information about the passable path using the test edges. It repeatedly traverse very expensive edges (e_{2m+n+i}) on the paths before realizing the next edge is not passable. This is because Optimistic assumes those edge are passable until it finds out that they are not. The value of α does not affect the average path cost because the cost of the shortest path on the determinized graph is usually much higher than the cost of the exploration path (often the path to the target is not present at all). UCT fails to improve the result of Optimistic with 100 and 300 rollouts. UCT relies heavily on Optimistic, which performs poorly in this experiment as a rollout policy and as a heuristic in guiding its search. In fact, UCT on this problem provides the only failures in the experiments. With 100 rollouts, it failed to reach the destination within 140 steps for 18 out of 500 runs.

In the second experiment, we only test on cases where the destination is reachable. The optimal strategy is nontrivial and depends on the sampled instances in the mixture model. However, there are instances where Optimistic is obviously sub-optimal. The road network is well-connected in general but there may be situations where there are long dead ends due to blockages. In such instances, Optimistic can incur significant cost traversing these dead ends because it does not reason about the uncertainty about upcoming edges. HSPD on the other hand is able to reason about uncertainty of edges ahead and may choose to gather information about potential dead ends before traversing them.

HSPD outperforms Optimistic for bigger values of α (4.0 and 8.0) but it under-performs Optimistic slightly for $\alpha = 1$. This is because it is often fairly cheap to visit a new location nearby to receive an observation that reduces version space by half, in comparison to reaching the goal by the shortest path. For this problem, the performance guarantee is very weak since the least likely configuration has very small probability. In spite of that, the algorithm is still practically useful as the scaling factor α helps to reduce over-exploration by making the exploration path look longer.

In this experiment, UCT improves the result of Optimistic significantly, by almost 25%. This is likely because Optimistic already has reasonable performance and UCT provides further improvements by searching in belief space and accounting for uncertainty in upcoming edges. However, its performance does not improve much when the number rollouts is increased from 100 to 300, and it still under-performs HSPD with $\alpha = 8.0$.

To apply the algorithm in practice, it would be useful to develop automatic methods for tuning α in each recursive call of HSPD. Computing the probability of reaching the goal with the exploitation path may provide useful information for doing that.

7 CONCLUSION

This paper introduces the Bayesian Canadian Traveler Problem. Compared with CTP, BCTP allows correlations among the unknown states of edges in a graph and provides a more realistic model for short-path finding in many applications. An optimal solution to BCTP requires careful balance between exploration and exploitation, a fundamental issue in reinforcement learning and planning under uncertainty.

We have developed Hedged Shortest Path under Determinization, a polynomial-time algorithm that approximates an optimal solution to BCTP within a polylogarithmic factor. It outperforms a widely used greedy algorithm and a state-of-the-art UCT-based search algorithm in our experiments. The basic ideas behind the algorithm are (i) the most likely edge determinization and (ii) hedging the exploration and exploitation paths. We believe that these ideas are useful in more general settings. A future research direction is to generalize the algorithm to model-based Bayesian reinforcement learning, where the transition dynamics is non-deterministic.

Acknowledgments. This work is supported in part by the Singapore MoE Tier 2 grant MOE2016-T2-2-068 and the US Air Force Research Laboratory under agreement number FA2386-15-1-4010.

References

[1] Amotz Bar-Noy and Baruch Schieber. The Canadian Traveller Problem. In *Proc. ACM-SIAM Symp. on Discrete Algorithms*, volume 91, pages 261–270, 1991.

[2] David Meir Blei and Leslie Pack Kaelbling. Shortest paths in a dynamic uncertain domain. In *Proc. IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*, volume 4, page 2, 1999.

[3] Zahy Bnaya, Ariel Felner, and Solomon Eyal Shimony. Canadian traveler problem with remote sensing. In *Proc. Int. Jnt. Conf. on Artificial Intelligence*, pages 437–442, 2009.

[4] Gruia Calinescu and Alexander Zelikovsky. The polymatroid steiner problems. *J. Combinatorial Optimization*, 9(3):281–294, 2005.

[5] Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *Proc. Uncertainty in Artificial Intelligence*, pages 150–159. Morgan Kaufmann Publishers Inc., 1999.

[6] Debadeepta Dey, Andrey Kolobov, Rich Caruana, Ece Kamar, Eric Horvitz, and Ashish Kapoor. Gauss meets Canadian traveler: shortest-path problems with correlated natural dynamics. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1101–1108. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[7] Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts Amherst, 2002.

[8] Patrick Eyerich, Thomas Keller, and Malte Helmert. High-Quality Policies for the Canadian traveler's problem. In *Proc. AAAI Conf. on Artificial Intelligence*, 2010.

[9] Dave Ferguson, Anthony Stentz, and Sebastian Thrun. PAO for planning with hidden state. In *Proc. IEEE Int. Conf. on Robotics & Automation*, volume 3, pages 2840–2847. IEEE, 2004.

[10] Dror Fried, Solomon Eyal Shimony, Amit Benbassat, and Cenny Wenner. Complexity of Canadian traveler problem variants. *Theoretical Computer Science*, 487:1–16, 2013.

[11] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems. In *Automata, Languages and Programming*, number 6198, pages 690–701. January 2010.

[12] Jörg Hoffmann. Simulated penetration testing: From "Dijkstra" to "Turing Test++". In *Proc. Int. Conf. on Automated Planning and Scheduling*, pages 364–372, 2015.

[13] Laurent Hyafil and Ronald L Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[14] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[15] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive informative path planning in metric spaces. In *Workshop on the Algorithmic Foundations of Robotics*, 2014.

[16] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive stochastic optimization: From sets to paths. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1576–1584, 2015.

[17] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Supplementary material for Adaptive stochastic optimization: From sets to paths. https://papers.nips.cc/paper/6005-adaptive-stochastic-optimization-from-sets-to-paths, 2015. NIPS Supplementary Material. Accessed: 2017-03-28.

[18] Evdokia Nikolova and David R Karger. Route planning under uncertainty: The Canadian Traveller Problem. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 969–974, 2008.

[19] Christos H Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

[20] Robert Platt, Russell Tedrake, Leslie Kaelbling, and Tomás Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In *Proc. Robotics: Science and Systems*, 2010.

[21] George Harry Polychronopoulos, John N Tsitsiklis, et al. Stochastic shortest path problems with recourse. 1993.

[22] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[23] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[24] Sung Wook Yoon, Alan Fern, and Robert Givan. FF-replan: A baseline for probabilistic planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*, volume 7, pages 352–359, 2007.