Exact Algorithms for the Canadian Traveller Problem on Paths and Trees

David Karger [*] Evdokia Nikolova [†]

July 6, 2007

Abstract

The Canadian Traveller problem is a stochastic shortest paths problem in which one learns the cost of an edge only when arriving at one of its endpoints. The goal is to find an adaptive policy (adjusting as one learns more edge lengths) that minimizes the expected cost of travel. The problem is known to be #P hard. Since there has been no significant progress on approximation algorithms for several decades, we have chosen to seek out special cases for which exact solutions exist, in the hope of demonstrating techniques that could lead to further progress. Applying techniques from the theory of Markov Decision Processes, we give an exact solution for graphs of parallel (undirected) paths from source to destination with random two-valued edge costs. We also offer a partial generalization to traversing perfect binary trees.

[*]MIT Computer Science & AI Lab, Cambridge MA 02139, USA, karger@csail.mit.edu
[†]MIT Computer Science & AI Lab, Cambridge MA 02139, USA, enikolova@csail.mit.edu

1 Introduction

The Canadian traveller problem was first defined by Papadimitriou & Yannakakis [13] to describe a situation commonly encountered by Canadian travellers: once a driver reaches an intersection, he sees whether the incident roads are snowed out or not and consequently decides which road to take. In a graph instance, we are given distributions for the costs of the edges, and upon arriving at a node we see the actual cost values of incident edges. The goal is to find an optimal *policy* (adaptive algorithm) for reaching from source S to destination T that minimizes expected cost.

The Canadian Traveller problem is exemplar of several interesting and important issues in optimization. It is a *stochastic optimization* problem, in which some details of the input (the edge lengths) are unknown when the problem is being solved. It is *adaptive*—decisions are made as the algorithm is running, based on new information that arrives during that time. There is a *cost to gather information*—one must travel to a vertex to discover its incident edge costs. It therefore involves the classic problem of *exploration versus exploitation* [17], in which one must decide whether to exploit the (possibly suboptimal) information acquired so far, or invest further cost in exploration in the hope of acquiring better information.

While it is a simple and elegant theoretical model, the Canadian traveller problem has been (independently) considered by different communities in addition to theoretical computer science such as electrical engineering [15] and artificial intelligence [2]. This problem and similar formulations finds application in transportation, planning, robot navigation and other areas. Its importance is convincingly demonstrated by the diversity of fields of research that have considered it, eg. [6, 2, 1].

Despite its importance, very little progress has been made in formal approaches to the problem. Canadian Traveller was shown to be #P-hard [13] via a reduction to the ST-reliability problem: given an ST reliability problem, a trivial construction yields a graph whose expected Canadian Traveller cost is equal to the probability the graph disconnects. While the connection is clear, there are also clear differences: ST reliability is easy to solve on series-parallel graphs, while no algorithm is known for Canadian Traveller on such graphs; conversely, Canadian Traveller is trivial on DAGs (as we show below) while no ST-reliability algorithm is known for such graphs.

Because of this negative result, the focus falls naturally on approximation algorithms. But to date, none have been developed. Indeed, even simpler questions such as whether there is a polynomial size *description* of an optimal or approximately optimal policy, regardless of whether it can be constructed, remain unanswered.

Our Results In light of the lack of progress on approximation algorithms, we have chosen to explore *exact* solution on special classes of graphs. Our aim is to begin to understand the structure of good solutions, and the barriers to finding them, as a first step towards more general instances. We explain the connection of our problem to *Markov Decision Processes (MDPs)*; a general model involving a *state space* and a set of *actions* that move you probabilistically from state to state and incur specified costs. The goal is to find a policy (choice of action at each state) that optimizes total cost. MDPs can be solved in time polynomial in the size of the state space. The problem is that for Canadian Traveller, the obvious state space is exponential in the size of the graph: there is one state for each possible set of (so far) observations of values of any reachable subset of the edges.

We begin with the (well understood) result that if incident edge costs are resampled each time we visit a vertex, the problem becomes a standard *Markov Decision Process* solvable in polynomial time. Similarly, in the case of a DAG, the problem is trivially solved by dynamic programming. In both cases, the state space is small because we need not remember edge—in the case of resampling, the edges will be different each time we look, and in the case of DAGs, we will never return to an edge. But as soon as one can return to edges of fixed value, this approach fails.

It became clear that a core question in Canadian traveller is when to turn back and seek a different route. To separate this question from the more general problem of which way to go forward, we considered the special case of a graph made of parallel paths. We give an exact algorithm for the case where the edges take on two distinct values at random. When one of the values is 0 the optimum policy has a simple closed form description; when both values are nonzero we show how to capture the problem in a *small* MDP. We then begin to extend our results to consider multiple ways forward: we give a closed form solution for a perfect binary tree whose edges are 0-1 random variables.

While these results are limited in scope we believe they are meaningful in light of the negligible progress on algorithms for Canadian Travelling. They demonstrates framings and techniques that may be useful for exact or approximation algorithms on more general classes of graphs.

Related Work Since each field has used a different term for the same problem such as the *bridge problem* [2] and others, related work is sometimes difficult to recognize. Among the work most closely related to ours after Papadimitriou & Yannakakis [13], is that of Bar-Noy and Schieber [1] who analyze an adversarial version of the problem from a worst-case perspective. The authors also consider a stochastic version, though overall their assumptions alter the nature of the problem significantly and allow for an efficient solution. We assume that once an edge cost is observed, it remains fixed for all time and remark in Section 2 on the much easier version in which a new value for the edge is resampled from its distribution at each new visit. Blei and Kaelbing [2] consider the same version of the Canadian traveller problem discussed here and offer solutions based on Markov Decision Processes, though they note that their solutions are exponential in the worst case.

A different perspective of adaptive routing problems is offered by the framework of competitive analysis [9]. Other work on adaptive routing includes [5, 7, 4, 8, 10, 14, 3], etc.

A key difficulty of the Canadian traveller problem is that it is adaptive. Since our objective is to minimize the expected cost of the route (namely the sum of the costs of its edges), in an offline setting the solution would trivially reduce to a deterministic shortest paths problem with edge weights equal to their expected cost (in contrast to a formulation where the cost of a path is a nonlinear function of the individual random edge costs [11, 12] and thus the offline solution is nontrivial or hard). On the other hand, although the objective here is linear, the ability to look one edge into the future implicitly brings about considerations of path dependence and variance disguised in the path expectation.

2 Problem formulation and Preliminaries

Consider a graph $G(V, E)$ with a specified source S and destination T. The edges in the graph have uncertain costs, coming from known probability distributions. For edge e, we shall denote its random variable cost by X_e and its realized value by x_e or simply $cost(e)$.[1] Upon reaching a node, an agent observes the realized values of all edges adjacent to that node. The objective is to find the optimal policy for routing from the source to the destination, which minimizes the expected cost of the route.

2.1 Background on Markov Decision Processes

We first briefly describe Markov Decision Processes (MDPs), which we will use in the following sections for the design and analysis of some of our algorithms.

[1]We distinguish between the cost of a path, which is the sum of the costs of its edges, and its length, which is the number of edges on the path.

In an MDP we are given a *state space* (each state comprising our knowledge of where we currently are and any other information we have), an *action space*, *probabilities of transitioning* from one state to another given an action, and a *cost* or reward which is a function of the state. The goal is to find a *policy*, that is a mapping of states to actions, that minimizes the expected cost of getting from the specified initial to the final state.[2] It is known that the optimal policy of MDPs can be found in polynomial time in the size of the MDP (the number of states and actions), for example via linear programming [16]. However, most often, the size of the MDP blows up and such exact solutions are typically exponential, unless there is a clever way of defining the MDP which reduces its size.

For our problem the optimal policy means the same as the optimal adaptive algorithm, which aims to get from the source to the destination via a route of least expected cost. Sometimes we also call that the optimal strategy.

The Canadian traveller problem can be reduced to a finite-state MDP in which a state is a node together with values of observed edges so far–note though that due to the unfolding history, even on a simple graph with Bernoulli edges this would lead to exponentially many states, and thus inefficient solution for the optimal policy.

2.2 An easy version: Canadian Traveller with resampling

Consider the Canadian Traveller problem as defined above with the additional assumption that every time we come back to a node, we observe newly resampled values for the incident edges. In this case, there a natural choice for an MDP that solves the problem in polynomial time.

Define an MDP in which is a state is simply the node of current position. An action is the choice of which next node to visit. Then the number of states is $|V|$, the number of vertices, and the MDP can be solved in polynomial time: in particular, we can find in polynomial time the expected cost $w(v)$ of the optimal policy for getting from each node v to the destination (prior to having seen the costs of edges incident to v). The optimal action once we arrive at a node v is then to choose its next neighbor v' minimizing the cost of edge (v, v') plus the optimal cost $w(v')$ to the destination.

2.3 Canadian Traveller on DAGs

For the rest of the paper, we consider the standard version of the Canadian Traveller problem, in which once observed, an edge cost remains fixed. In this section we show that the special case of directed acyclic graphs (DAGs) can be solved in polynomial time.

In general the Canadian Traveller problem with fixed observations is much harder than the version with resampling: perhaps surprisingly, since in essence we are now better informed of the costs of all previously unobserved edges. From the point of view of MDPs, a corresponding MDP modeling this version requires a much bigger state space since a state needs to reflect all past observations. An exact solution would therefore require exponential time.

In a DAG on the other hand, we can never return to an edge that has been previously visited, so the versions with and without resampling are essentially identical problems. Thus we can solve DAGs in polynomial time with the same MDP as in Section 2.2. We now give a direct faster polynomial-time solution (in the Appendix) for general DAGs based on dynamic programming.

Theorem 1. *The Canadian traveller problem on a directed acyclic graph with $|E|$ edges can be solved exactly in time $O(|E|)$.*

[2]For more background on MDPs, see *e.g.*, Puterman [16].

3 Properties of Disjoint-Path Graphs

In this section, we derive a key monotonicity property of the optimal policy on undirected disjoint-path graphs (consisting of disjoint ST-paths), with independent identically distributed edges.

Consider an undirected graph with source S, destination T, consisting of k node-disjoint paths between S and T. Any algorithm would follow one path up to a point, then possibly return through the source to explore a part of the second path, etc. until it reaches the destination. In the worst case, it may turn at every node, reaching the next unexplored node on another path, then turning back to reach the next unexplored node on a third path, etc. At every step of the algorithm we arrive at a configuration where we know the distance ahead of us a_i and n_i unexplored edges on the i-th path for $i = 1, ..., k$, as shown in Figure 1. In this configuration, we denote the expected cost to reach the destination under the optimal algorithm by $w(\{a_i, n_i\}_{i=1}^k)$. With a slight abuse of notation, we will use the same notation for a configuration with current position inside a path. Then for a different path i, a_i will stand for the sum of distances from the current location back to the source and from the source to the first unexplored edge on path i.

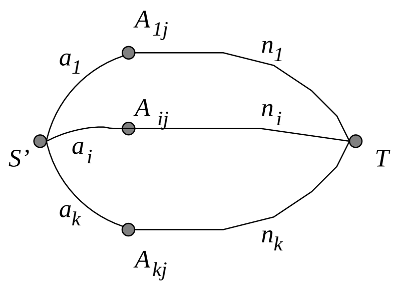

Figure 1: $w(a, b, m, n)$ is the optimal expected cost to get from S' to T.

Lemma 2. *[Monotonicity] The following properties hold for the optimal algorithm and its expected cost* $w(\{a_i, n_i\}_{i=1}^k)$*:*

1. *The optimal expected cost is symmetric with respect to the paths,* $w(a_i, n_i, a_j, n_j, ...) = w(a_j, n_j, a_i, n_i, ...)$.
2. *The optimal expected cost is monotone nondecreasing in* a_i *and* n_i*, for all* $i = 1, ..., k$.

Proof. Part (1) follows from the isomorphism of the graphs with the ith and jth paths exchanged.

For part (2), we first show that the optimal expected cost is monotone nondecreasing in n_i. By symmetry, it suffices to show this for n_1. We need to show $w(a_1, n_1, \{a_j, n_j\}_{j\neq 1}) \leq w(a_1, n_1 + n, \{a_j, n_j\}_{j\neq 1})$ for $n > 0$. Denote the configurations $\mathcal{C}_{n_1} = (a_1, n_1, \{a_j, n_j\}_{j\neq 1})$, $\mathcal{C}_{n_1+n} = (a_1, n_1 + n, \{a_j, n_j\}_{j\neq 1})$ and $\tilde{\mathcal{C}}_{n_1+n} = (a_1, n_1 + n, \{a_j, n_j\}_{j\neq 1})$ where the last n edges on the first path have cost 0.

Suppose the optimal algorithm on $\mathcal{C}_{n_1+n}$ is $\mathcal{A}$. Consider algorithm $\mathcal{A}^*$ on configuration $\tilde{\mathcal{C}}_{n_1+n}$ defined as follows:
(i) Run $\mathcal{A}$ until it reaches the destination T or the node T' on the first path, which immediately precedes the n zero edges;
(ii) If reach T before T', terminate;
(iii) Else, proceed from T' to T along the bottom edges at cost 0.

On every realization of the edge random variables in configuration $\mathcal{C}_{n_1+n}$, Algorithm $\mathcal{A}^*$ incurs the same or smaller cost $\tilde{\mathcal{C}}_{n_1+n}$ than Algorithm $\mathcal{A}$ on $\mathcal{C}_{n_1+n}$. On the other hand, the cost of running $\mathcal{A}^*$ on configuration $\tilde{\mathcal{C}}_{n_1+n}$ is precisely the same as the cost of running $\mathcal{A}$ on configuration $\mathcal{C}_{n_1}$. Therefore,

$$\begin{aligned} w(a_1, n_1, \{a_j, n_j\}_{j\neq 1}) &\leq w_{\mathcal{A}}(a_1, n_1, \{a_j, n_j\}_{j\neq 1}) \\ &= w_{\mathcal{A}^*}(\tilde{\mathcal{C}}_{n_1+n}) \\ &\leq w(a_1, n_1+n, \{a_j, n_j\}_{j\neq 1}). \end{aligned}$$

We can show similarly that the optimal expected cost is monotone non-decreasing in a_i for all $i = 1, ..., k$. We explain this briefly for a_1. Denote configuration $\mathcal{C}_{a_1} = (a_1, n_1, \{a_j, n_j\}_{j\neq 1})$, $\mathcal{C}_{a+a_1} = (a + a_1, n_1, \{a_j, n_j\}_{j\neq 1})$ for some $a > 0$. Run the optimal algorithm $\mathcal{A}$ for $\mathcal{C}_{a+a_1}$ on this configuration, except when we record the cost it incurs, we record 0 whenever it traverses the links of total distance a along the first path—call this algorithm $\mathcal{A}^*$. Clearly then, the expected cost $w_{\mathcal{A}}(\mathcal{C}_{a+a_1}) \geq w_{\mathcal{A}^*}(\tilde{\mathcal{C}}_{a_1})$, hence

$$\begin{aligned} w(a_1, n_1, \{a_j, n_j\}_{j\neq 1}) &\leq w_{\mathcal{A}^*}(\tilde{\mathcal{C}}_{a_1}) \\ &\leq w_{\mathcal{A}}(\mathcal{C}_{a+a_1}) \\ &= w(a + a_1, n_1, \{a_j, n_j\}_{j\neq 1}), \end{aligned}$$

this concludes the proof. □

Theorem 3. *Suppose $a_1 = \min a_i$ and $n_1 = \min n_i$. Suppose also that the cost of every edge is a non-negative random variable. Then it is optimal to proceed along the first path up to the first unseen edge on the path.*

Proof. We first show this for the special case $n_i = n$ for all $i = 1, ..., k$. We have two choices, routing along the first path or along the ith path, for some $i \neq 1$. In the first case, the expected cost under an optimal consequent strategy would be $a_1 + \mathbf{E}_L\, w(\{L, n-1\}, \{a_i + a_1, n\}, \{a_j + a_1\}_{j\neq 1,i})$. In the second case, the expected cost is

$$\begin{aligned} &a_i + \mathbf{E}_L\, w(\{L, n-1\}, \{a_1 + a_i, n\}, \{a_j + a_i\}_{j\neq 1,i}) \\ &\geq a_1 + \mathbf{E}_L\, w(\{L, n-1\}, \{a_1 + a_i, n\}, \{a_j + a_1\}_{j\neq 1,i}), \end{aligned}$$

since $w(\{x, n-1\}, \{a_1 + a_i, n\}, \{a_j + a_i\}_{j\neq 1,i}) \geq w(\{x, n-1\}, \{a_1 + a_i, n\}, \{a_j + a_i\}_{j\neq 1,i})$ for every x by the symmetry and monotonicity of $w(.)$. Thus, it is always optimal to go along the first path, and it is strictly optimal if $a_1 < a_i$ for all $i \neq 1$.

Next, we show the statement for general n_i. Since $n_i \geq n_1$, for all i and every realization of the first $n_i - n_1$ edges along the ith path the resulting configuration would be $(a_1, n_1, \{a_i', n_1\}_{i>1})$ where $a_i' > a_i$, by nonnegativity of the edges. Thus, $a_1 \leq a_i'$ for all i so in every realization it is optimal to proceed along the first path. Hence in the original configuration $(\{a_j, n_j\}_{j=1}^{k})$ it is optimal to proceed along the first path, which concludes the proof. □

4 Optimal policy on Disjoint-path graphs

4.1 $(0, 1)$ Bernoulli edges

In this section, we compute exactly the optimal cost when the edges have cost 0 with probability p and 1 (equivalently a) otherwise.

Theorem 3 immediately establishes that the optimal strategy explores all paths until it reaches edges of cost 1 on each path, at which point it comes back to the path with fewest unexplored edges and follows it until the end.

Theorem 4 (Optimal Adaptive Algorithm). *Suppose we have a graph with k parallel paths and edges with independent identically distributed edge costs 0 or 1. The optimal strategy explores all 0 edges at the beginning of each path until it reaches a 1-edge on the path (or the destination), then returns to the path with fewest unexplored edges and continues without turning to the end of the path.*

With this, we can compute exactly the expected cost of the optimal route. The case of general length paths is given in Theorem 14 in the Appendix and the following corollary follows from it.

Corollary 5. *When the graph consists of k undirected disjoint paths with n edges each, and all edges are independent and identically distributed $(0,1)$ Bernoulli variables, the expected cost of the optimal strategy is*

$$\sum_{i=0}^{n-1} \left[(1-p^{n-i})^k - (1-p^{n-i-1})^k\right]\Big(1 + i\,\mathbf{E}[X]\Big),$$

where $\mathbf{E}[X] = 1-p$ is the mean of a single edge.

4.2 Disjoint-path graphs with positive Bernoulli edges

In this section we compute an optimal policy based on MDPs for general positive Bernoulli edges, without loss of generality taking value 1 with probability p and value K otherwise. Defining MDPs in the natural way yields exponentially large state space as explained in Section 2; combining it with the special structure of the optimal policy here lets us define a more parsimonious MDP that is efficiently solvable.

A property of the optimal policy here is that once we have crossed a K-edge, we will never cross it again. This follows from a distribution-dominance argument, somewhat similar to the monotonicity lemma in Section 3. Similarly, once we have chosen to follow a path, the optimal policy would dictate to keep following it until the first K-edge.

Lemma 6. *Suppose the optimal policy chooses a path out of several node-disjoint paths from the source to the destination. Then it is optimal to follow this path until seeing the first cost-K edge on the path. Further, once the optimal algorithm crosses a cost-K edge, it is optimal to continue along the same path until the destination.*

Proof. (Sketch) When the optimal policy chooses a particular action (path), it does so assuming that the cost 1 of crossing the current edge plus the expected cost of the optimal policy at the next node dominates (is no greater than) those of the remaining actions. If the next edge on the current path is also 1, this is the best possible scenario, which would decrease the expected cost of continuing along the path and dominate the action of turning back and proceeding along a different path. Thus, it is optimal to proceed along the current path at least until seeing a cost-K edge.

Further, suppose the optimal policy has crossed a K-edge. Again the action of continuing along the current path dominates the action of turning back since the latter would now incur cost at least K greater than prior to crossing the K-edge, and the configuration (cost of next edge plus remaining unseen edges) along the current path still dominates the configuration on the remaining paths. $\square$

The properties of the optimal policy allow for a very simple description of it: explore one or more of the paths up to the first cost-K edge along them, then pick one and follow it until the end. The policy just has to decide in what order to explore the paths, and how many, before committing to a path. This allows us to capture it as the solution of an MDP with concise representation.

4.2.1 Two paths

In order to solve the general case, the optimal policy needs a subroutine for comparing two paths.

Lemma 7. *The optimal policy on a graph with two node-disjoint paths and positive Bernoulli edges can be found in polynomial time.*

Proof. At each instant prior to committing to a path, our knowledge on the graph can be described concisely as the tuple $(a_1, x_1, n_1; a_2, x_2, n_2; i)$, where a_i is the number of cost-1 edges we have observed along path i; $x_i = 1$ or K is the last observation on that path; n_i is the number of unobserved edges remaining on the path and $i = 1, 2$ is the index of the current path. Clearly these are polynomially many $O(|E|^4)$ states and in each state there are only two actions: to continue along the current path or turn back to the other path. ($|E|$ is the number of edges in the graph, as usual.) Thus, the optimal policy can be obtained as the solution to that MDP in polynomial time. $\square$

As a corollary from the two paths case, we have an efficient subroutine for comparing two paths with configurations (a_i, x_i, n_i) using the notation above, in terms of which one would yield a lower expected policy cost.

4.2.2 Arbitrary many paths

We can now derive the optimal policy for the general case of more than two paths.

Theorem 8. *The optimal policy on a disjoint-path graph with positive Bernoulli edges can be found in polynomial time.*

Proof. Using the subroutine for two paths, as soon as we explore more than two paths in the given graph, it suffices to keep information about the best explored path so far.

Initially, order the paths which start with a cost-1 edge in order of increasing length. By dominance, it would be optimal to explore them in this order. If there is more than one path starting with K, keep only the one with fewest edges (since by dominance, we would choose to explore this path before the others starting with K, but once we cross a K, we would commit to the first such path).

We have two possibilities. If all paths start with K, by dominance it is optimal to commit to the path with fewest edges. Otherwise, at least one path starts with 1 and we proceed as follows. Consider an MDP in which the state is given by the tuple $(a_1, x_1, n_1; a_2, x_2, n_2; i)$ where a_1 is the number of ones we have observed along the best explored path so far, $x_1 = K$ if the path has been explored ($x_1 = 1$ only initially), n_1 is the number of unobserved edges along the best path; (a_2, x_2, n_2) are the corresponding values of the current path i. Again, there are at most $O(|E|^5)$ states in the MDP; for each state there are three possible actions: continue along the current path, go back to the best previously explored path or continue to the next unexplored path. Thus, the optimal policy can be found in polynomial time. $\square$

5 Series-parallel graphs

In this section we show that we may sometimes turn back along a cost 1 edge in a series-parallel graph, so the optimal policy for disjoint-path graphs no longer applies here.

Consider the series-parallel graph in Figure 2, in which an agent starting at S has a choice of two directions, to A or B. The graph from A to T consists of n paths with $\log n$ edges each, and there is a single edge from B to T. Further suppose that each edge is 0 or 1 with probability $\frac{1}{2}$ each.

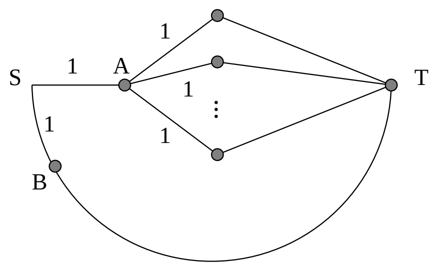

Figure 2: The optimal policy may return along a traversed edge of cost 1.

Lemma 9. *The optimal routing strategy on the given graph configuration first goes to A, and upon observing all cost 1 edges at A, goes back to S, then towards B and T.*

Proof. First consider only the disjoint-path graph between A and T. Since it consists of n paths with $\log n$ edges each, the expected cost of the optimal algorithm on this graph is

$$\sum_{i=0}^{\log n-1}\left[(1-p^{\log n-i})^n-(1-p^{\log n-i-1})^n\right]\left(1+\frac{i}{2}\right)\leq .45$$

for $p=\frac{1}{2}$ and $n\geq 5$. Therefore the expected cost of the optimal strategy at A, including the possibility of going back and using the path through B, is no more than $.45$. On the other hand, once we reach B, it would be optimal to continue until T and so the expected cost at B is the expected cost of edge BT, that is $.5$.

Hence it is initially optimal to go to A. However upon seeing that all incident edges at A have cost 1, the optimal cost of continuing one more step along one of the paths is at least $1+\frac{4}{2}=3$, assuming we have $(\log n)\geq 5$ edges per path. That is because each path with $(\log n)$ edges from A to T is weakly dominated by the path we proceed along, as in the proof of the Monotonicity Lemma. Hence we would either proceed until the end of the path at expected cost of at least 2, or turn back to A, S, B at cost at least 3.

On the other hand, the expected optimal cost of returning from A to S, B, T is 2.5. Therefore, upon seeing all 1-edges at A, it is optimal to cross back edge AS and proceed to T via B. ☐

6 Trees with $(0,1)$ Bernoulli edges

The results in the previous sections suggest that the Canadian Traveller problem has an inherent difficulty when subsequent paths can cross and have dependent values; the design of the algorithms also crucially revolves around branching inside the graph—the points where we decide to continue or turn back. This leads us to examine trees as a natural generalization of graphs with disjoint paths, where the goal is to reach from the root to *any* leaf while minimizing the expected cost of the route.

In this section, we compute the optimal policy efficiently for perfect binary trees with $(0,1)$ Bernoulli edges and show that as in series-parallel graphs, general trees unfortunately do not conform to the same simple form of optimal policy.

6.1 Optimal policy on perfect binary trees

Consider perfect binary trees with Bernoulli edges which are 1 with probability p and 0 otherwise. As in the case of parallel paths, it is optimal to explore all 0 edges out of the root; if we reach a leaf in this way, we

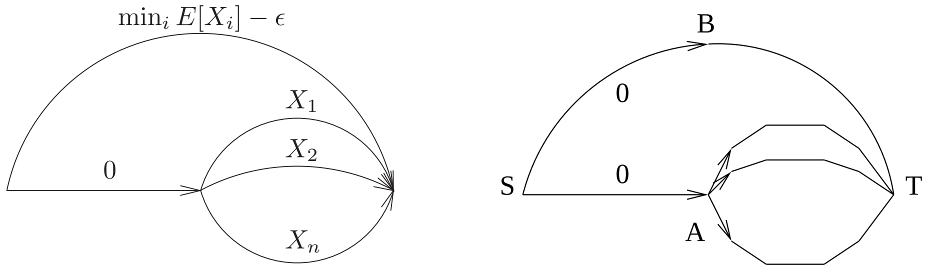

Figure 3: Gap from the optimal policy by: (left) the minimum expected distance heuristic and (right) the expected minimum distance heuristic.

are done. Otherwise, the question is which 1-edge to cross. By a similar dominance argument as in the case of disjoint paths, it is optimal to commit to the branch of the tree of smallest depth of unexplored edges. In addition, by the same dominance argument it is never optimal to cross back a cost 1-edge. Finally, once we commit to a sub-branch of the tree, the optimal policy repeats an exploration of all 0-edges, commits to crossing a 1-edge leading to a further subbranch of smallest depth, etc.

Theorem 10. *The optimal policy on perfect binary trees can be computed in polynomial time.*

6.2 Nonoptimality for general trees

Unfortunately the simple optimal policy above does not extend to more general trees with $(0, 1)$-Bernoulli edges. A similar counterexample to the series-parallel graph in Section 5 shows that the optimal policy may sometimes cross back on a cost-1 edge (consider the tree corresponding to the series-parallel graph in Figure 2 in which the different paths branch out into leaves rather than ending at the destination T). It remains open to see whether a polynomial-size MDP exists to find the optimal policy efficiently.

7 Adaptive Heuristics

In the absence of an efficient algorithm for the Canadian Traveller problem on general graphs, we examine two natural heuristics for the optimal routing policy. Their names reflect precisely what they do: the *minimum expected distance* heuristic chooses to go to the next node, such that the expected distance from the current node to the destination is minimized. On the other hand, the *expected minimum distance* heuristic goes to the next node, minimizing the sum of the current edge cost and the expected minimum distance from the next node to the destination.

7.1 Minimum Expected Distance Heuristic

We will show that the Minimum Expected Distance Heuristic has an exponential gap from the optimal routing algorithm, that is it may result in a route with an exponentially higher cost than that of the optimal route, and should therefore be avoided.

Consider the graph in Figure 3. Any route from the source to the destination consists of the direct link on the top or of the 0-cost link on the bottom and one of the subsequent n parallel links, whose costs are given by the random variables $X_1, ..., X_n$. Suppose the direct link on the top has cost $\min_i E[X_i] - \epsilon$.

Suppose X_i are Bernoulli random variables, which are 1 with probability $p > 0$ and 0 otherwise. If at least one of the n links on the bottom is 0, the bottom route would have cost 0, otherwise it would have cost

1. Thus, $\mathbf{E}[\min X_i] = p^n$. On the other hand, if we follow a partially adaptive algorithm which replaces the unseen edges with their expectation, we would take the top route and the expected cost of the trip would be $\min \mathbf{E}[X_i] - \epsilon = p - \epsilon$. With $\epsilon \to 0$, we see that this gap is exponential:

$$\frac{\min \mathbf{E}[X_i] - \epsilon}{\mathbf{E}[\min X_i]} = \frac{p}{p^n} = \frac{1}{p^{n-1}}.$$

Lemma 11. *The minimum expected distance heuristic has exponential gap from the optimal policy.*

7.2 Expected Minimum Distance Heuristic

In this section we show that the expected minimum distance heuristic may yield $(\log n)$-suboptimal routes, although it is significantly better and coincides with the optimal policy on disjoint-path graphs. Note that the expected minimum distance can be approximated arbitrarily well in polynomial time via sampling.

Lemma 12. *The expected minimum distance rule is optimal on disjoint-path graphs.*

Proof. See Appendix. □

Lemma 13. *The expected minimum distance heuristic has $\Omega(\log n)$ gap from the optimal policy.*

Proof. (Sketch) Consider the graph on the right in Figure 3, in which there are n paths of $(\log n)$ independent identically distributed edges each from A to T. With this, the minimum distance heuristic would prefer A to B since it is highly likely that there is a zero-cost route from A to T. However this route cannot be found in practice, and the expected cost of the bottom route is $(\log n)$, while that of the top route is constant. The optimal policy would naturally select the top route, and thus the gap of the expected minimum distance heuristic to the optimal policy can be as high as $\Omega(\log n)$. □

8 Conclusion

We have considered the Canadian traveller problem, in which the goal is to find an optimal adaptive algorithm for reaching from a source to a destination, when edges have stochastic (Bernoulli) costs and are revealed upon reaching an incident vertex, prior to traversing the edge. This slight change from traditional stochastic routing problems, giving an ability to look ahead, makes the problem particularly hard in that even minimizing a linear function—the expected cost of the route, implicitly involves the variance of costs and path dependence. Nevertheless, we identify classes of graphs on which the problem can be solved efficiently. Our solutions are based on a combination of structural properties of the graphs and optimal policies, and Markov decision processes.

Since the problem is in general hard, it remains open to see if the graphs we consider can be extended to a more general class that admits exact polynomial-time solutions, or otherwise to provide hardness results for generalizations of these classes, for example for series-parallel graphs. Given the importance of the problem in practice, it is also key to understand the effectiveness and limitations of the heuristics considered here and others, in theory and practice.

The current status of the general Canadian traveller problem is still widely open with respect to approximability. Perhaps related to the latter, it would be intriguing to understand the mutual relationship and complexity between the Canadian traveller and the ST-reliability problem.

Acknowledgement

We thank Shuchi Chawla, Nick Harvey, David Kempe, Vahab Mirrokni, Christos Papadimitriou, John Tsitsiklis and Mihalis Yannakakis for valuable discussions.

References

[1] A. Bar-Noy and B. Schieber. The canadian traveller problem. In *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, pages 261–270, 1991.

[2] D. Blei and L. Kaelbling. Shortest paths in a dynamic uncertain domain. In *IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*, 1999.

[3] A. Blum, E. Even-Dar, and K. Ligett. Routing without regret: On convergence to nash equilibria of regret-minimizing algorithms in routing games. In *Proceedings of PODC*, 2006.

[4] J. Boyan and M. Mitzenmacher. Improved results for route planning in stochastic transportation networks. *Symposium of Discrete Algorithms*, 2001.

[5] Y. Fan, R. Kalaba, and I. J. E. Moore. Arriving on time. *Journal of Optimization Theory and Applications*, forthcoming.

[6] D. Ferguson, A. Stentz, and S. Thrun. PAO* for planning with hidden state. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[7] S. Gao and I. Chabini. Optimal routing policy problems in stochastic time-dependent networks. In *Proceedings of the IEEE 5th International Conference on Intelligent, Transportation Systems*, pages 549–559, Singapore, 2002.

[8] M. T. Hajiaghayi, R. D. Kleinberg, F. T. Leighton, and H. Raecke. New lower bounds for oblivious routing in undirected graphs. In *Proceedings of Symposium on Discrete Algorithms*, 2006.

[9] A. Kalai and S. Vempala. Efficient algorithms for on-line optimization. *Journal of Computer and System Sciences*, 71:291–307, 2005.

[10] A. B. McDonald. Survey of adaptive shortest-path routing in dynamic packet-switched networks. citeseer.ist.psu.edu/mcdonald97survey.html.

[11] E. Nikolova, M. Brand, and D. R. Karger. Optimal route planning under uncertainty. In *Proceedings of International Conference on Automated Planning and Scheduling*, 2006.

[12] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher. Stochastic shortest paths via quasi-convex maximization. *Proceedings of European Symposium of Algorithms*, 2006.

[13] C. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.

[14] L. Peshkin and V. Savova. Reinforcement learning for adaptive routing. In *Proceedings of Intnl. Joint Conf. on Neural Networks, IJCNN*, 2002.

[15] G. Polychronopoulos and J. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27(2):133–143, 1996.

[16] M. L. Puterman. *Markov Decision Processes*. John Wiley and Sons, 1994.

[17] R. Sutton and A. Barto. Reinforcement learning: An introduction, 1998. http://www.cs.ualberta.ca/ sutton/book/ebook/node7.html.

Appendix

Proof of Theorem 1. Denote by $w(v)$ the expected cost of following the optimal policy from node v to the destination, before we have reached v (*i.e.*, we have not observed the costs of the edges outgoing from v). The algorithm traverses the nodes of the graph in reverse topological order and computes $w(v)$ for each node v as

$$w(v) = \mathbf{E}[\min_{v'}\{X_{vv'} + w(v')\}],$$

where $X_{vv'}$ is the random cost of edge (v, v'). The calculation of a minimum of several random variables is standard in statistics. In the graph, we can compute the expected minimum by sampling or, if the edge costs have finite constant support, then we can compute it directly as $p_1c_1 + (1-p_1)[p_2c_2 + (1-p_2)[p_3c_3 + ...]]$, where $c_1 < c_2 < ...$ are the possible values of $[X_{vv'} + w(v')]$ and $p_1, p_2, ...$ their corresponding probabilities.

Having computed the optimal policy costs $w(v)$ for every node v, the optimal adaptive algorithm upon reaching a node v chooses to go to its neighbor v' minimizing $cost(v, v') + w(v')$. Since we go through each edge no more than twice, the total running time of the algorithm is $O(|E|)$.

The optimality of this algorithm critically follows from the fact that the graph is directed and acyclic, thus the optimal policy cost $w(v)$ at every node v only depends on the children of v and hence it is computed correctly (proof by induction, starting from the last node). Since our goal is to minimize the expected cost of the route, the optimal action at each node v would be to go next to the neighbor v' minimizing the sum of the cost of edge (v, v') and the optimal cost $w(v')$ from v' to the destination. $\square$

Theorem 14. *When the graph consists of k undirected disjoint paths with $n_1 \leq ... \leq n_k$ edges respectively, and all edges are independent and identically distributed Bernoulli (a, p) variables, the expected cost of the optimal strategy is*

$$\sum_{i=0}^{n_1-2}\Big[\prod_{j=1}^{k}(1-p^{n_j-i}) - \prod_{j=1}^{k}(1-p^{n_j-i-1})\Big]\Big(a + i\,\mathbf{E}[L]\Big)$$
$$+(1-p)^m\Big[\prod_{j=m+1}^{k}(1-p^{n_j-n_1-1}) -$$
$$\prod_{j=m+1}^{k}(1-p^{n_j-n_1})\Big]\Big(a + (n_1-1)\,\mathbf{E}[L]\Big),$$

where $\mathbf{E}[L] = a(1-p)$ is the mean of a single edge and $m \leq k$ is the number of paths with n_1 edges.

Proof. According to the optimal strategy, we explore each path until we reach the destination or the first edge of cost a and then proceed along the path with fewest unexplored edges, until the end. If some path has all zero edges until the end, the cost of the algorithm is zero. Otherwise, the cost is $(a + i\,\mathbf{E}[L])$ since we cross one edge a and then follow i edges of mean $\mathbf{E}[L]$ each until the end.

Now we just need to compute the probability that there are $i \leq 0$ unexplored edges along the path with fewest unexplored edges. There are at least i unexplored edges after the first edge a on path j if there is at least one edge a among the first $n_j - i$ edges; this happens with probability $(1-p^{n_j-i})$. Thus, the probability that there are at least i unexplored edges on the path with fewest unexplored edges, is given by

$$\begin{aligned} & \mathbf{Pr}(\text{\# unexplored edges } \geq i) \\ = \ & \mathbf{Pr}(\text{on each path } j, \text{ the first } n_j - i \\ & \text{ edges have least least one } a) \\ = \ & \prod_{j=1}^{k} (1 - p^{n_j - i}). \end{aligned}$$

Now, the probability that there are exactly $i = 0, 1, ..., n_1 - 2$ unexplored edges on the path with fewest unexplored edges, is

$$\begin{aligned} & \mathbf{Pr}(\text{\# unexplored edges } = i) \\ = \ & \mathbf{Pr}(\text{\# unexplored edges } \geq i) \\ & - \mathbf{Pr}(\text{\# unexplored edges } \geq i+1) \\ = \ & \prod_{j=1}^{k} (1 - p^{n_j - i}) - \prod_{j=1}^{k} (1 - p^{n_j - i - 1}). \end{aligned}$$

There are exactly $i = n_1 - 1$ unexplored edges when there is an a edge at the beginning of each path with n_1 total edges (say there are $m \leq k$ such paths, *i.e.*, that $n_1 = ... = n_m < n_{m+1} \leq ... \leq n_k$), and there are at least $n_1 - 1$ unexplored edges on each of the remaining paths. The probability that there are $n_1 - 1$ unexplored edges is therefore

$$\begin{aligned} & \mathbf{Pr}(\text{\# unexplored edges } = n_1 - 1) = \\ & (1-p)^m \Big[\prod_{j=m+1}^{k} (1 - p^{n_j - n_1 - 1}) - \prod_{j=m+1}^{k} (1 - p^{n_j - n_1}) \Big]. \end{aligned}$$

and the result follows. □

Proof of Lemma 12. This can be verified by examining what the two algorithms do in the three possible situations:

(i) Top first edge is 1, bottom first edge is 0. Say the next node on the top is A and the next node on the bottom is B. Since at B we can reach the destination by going back through S and A, $\min distance(B,T) \leq 1 + \min distance(A,T)$, therefore $\mathbf{E}[\min distance(B,T)] \leq 1 + \mathbf{E}[\min distance(A,T)]$ so the expected minimum distance algorithm will choose to go to B next, as will the optimal algorithm.

(ii) Top first edge is 1, bottom first edge is 1. Then $\mathbf{E}[\min distance(B,T)] \leq \mathbf{E}[\min distance(A,T)]$ whenever there are fewer edges from B to T hence the two algorithms again coincide.

(iii) We have crossed an edge with value 1 and we have to show that the expected minimum distance algorithm will continue along the corresponding path until the destination. This reduces to the two cases above.

□