Canadian Traveler Problem with Remote Sensing

Zahy Bnaya
Information Systems Engineering
Deutsche Telekom Labs
Ben-Gurion University
Be'er-Sheva, Israel
zahy@bgu.ac.il

Ariel Felner
Information Systems Engineering
Deutsche Telekom Labs
Ben-Gurion University
Be'er-Sheva, Israel
felner@bgu.ac.il

Solomon Eyal Shimony
Computer Science
Deutsche Telekom Labs
Ben-Gurion University
Be'er-Sheva, Isrel
shimony@cs.bgu.ac.il

Abstract

The Canadian Traveler Problem (CTP) is a navigation problem where a graph is initially known, but some edges may be blocked with a known probability. The task is to minimize travel effort of reaching the goal. We generalize CTP to allow for remote sensing actions, now requiring minimization of the sum of the travel cost and the remote sensing cost. Finding optimal policies for both versions is intractable. We provide optimal solutions for special case graphs. We then develop a framework that utilizes heuristics to determine when and where to sense the environment in order to minimize total costs. Several such heuristics, based on the expected total cost are introduced. Empirical evaluations show the benefits of our heuristics and support some of the theoretical results.

1 Introduction.

Efficient navigation to a predefined goal in unknown, partially known, or dynamically changed environments is a fundamental field of research in AI, robotics and other areas of computer science. While *navigation* is a common term, different assumptions can be made about the available actions of the agent and about the ways that the agent discovers information about unknown parts of the environment. The cost function to be minimized can also vary.

Basic actions generally assumed are *move* actions, where the agent moves from its current location (cell in a gird, vertex in a graph) to a neighboring location. Almost all navigation algorithms assume that a move of the agent incurs a cost (usually proportional to the length of the move) called *travel cost*. One usually assumes that a navigating agent can sense its immediate local environment (adjacent cells in a grid, outgoing edges in a graph etc.); called *local sensing* in this paper. Local sensing is usually assumed to be performed at no cost.

Additionaly, in some scenarios an agent can activate a sensor towards a distant location, (e.g. in a graph, towards specific vertex or edge, or even a given area of the graph). We call this *remote sensing*. Some prior work assume a static environment and no remote sensing abilities [Nikolova and Karger, 2008; Papadimitriou and Yannakakis, 1991; Bar-Noy and Schieber, 1991; Felner *et al.*, 2004]. Others assume dynamic environments, allow for remote sensing, but assume that *all* changes are immediately visible to the agent at no cost [Stentz, 1994; Koenig and Likhachev, 2002].

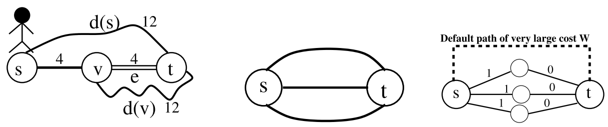
In this paper, the agent's task is to reach the goal while aiming to minimize a total cost function including **both** sensing and traveling. Specifically, we focus on the Canadian Traveler Problem (CTP) - a special kind of navigation problem. We generalize this problem to the case where remote sensing is allowed at a given cost. Both versions (with and without remote sensing) are intractable. We provide efficient optimal solutions for special case graphs. We then develop a framework that utilizes heuristic functions to determine when and where to sense the environment in order to minimize total cost. We develop several such heuristics and provide experimental results that show their benefit.

2 The Canadian Traveler Problem

In the *Canadian traveler problem*(CTP) [Papadimitriou and Yannakakis, 1991] a traveling agent is given a connected weighted graph $G = (V, E)$ as input together with its initial source vertex ($s \in V$), and a target vertex ($t \in V$). The input graph G may undergo changes, that are not known to the agent, *before* the agent begins to act, but remains fixed subsequently. In particular, some of the edges in E may become *blocked* and thus untraversable. Each edge e in G has a weight, or cost, $w(e)$, and is blocked with a probability $p(e)$, where $p(e)$ is known to the agent.[1] The agent can perform *move* actions along an unblocked edge which incurs a *travel cost* $w(e)$. Traditionally, the CTP was defined such that the status of an edge can only be revealed apon arriving at a node incident to that edge, i.e., only *local sensing* is allowed. In this paper we call this variant the *basic* CTP variant. The task of the agent is to travel from s to t while aiming to minimize the total travel cost C_{travel}. As the exact travel cost is uncertain until the end, the task is to devise a traveling strategy which yields a small (ideally optimal) *expected* travel cost.

We generalize the CTP and introduce the *CTP with sensing* variant. In this variant, in addition to move actions (and local sensing), an agent situated at a vertex v can also perform a *Sense* action and query the status of any edge $e \in G$. This action is denoted $sense(v, e)$, and incurs a cost $SC(v, e)$, or just $SC(e)$ when the cost does not depend on v. The cost function is domain-dependent, as discussed below. The task of the agent is to travel to the goal while minimizing a total cost $C_{total} = C_{travel} + C_{sensing}$.

[1]Note that it is sufficient to deal only with blocking of *edges*, since a blocked vertex would have all of its incident edges blocked.



(a) Expected cost (b) Disjoint paths (c) Special case

Figure 1: Navigation and the expected cost heuristic

To illustrate CTP with sensing, consider the simple example from figure 1.a where all edges are known to be traversable, except for edge e. If e is traversable, the cheapest path is (s, v, t) and its cost is 8. If e is blocked, the cheapest path is $d(s)$ and its cost is 12. The agent may choose not to sense edge e from s and to take the risk and travel to v. If it then realizes (from v with local sensing) that e is traversable, the travel cost will be 8. However, if it finds e to be blocked it will need to use segment $d(v)$ with an additional cost of 12 and the total travel cost will be 16. Alternatively, the agent might choose to sense e from s, and pay the sensing cost. Then, if e is traversable the total cost will be 8 plus the sensing cost and if e is blocked, the total cost will be 12 (segment $d(s)$) plus the sensing cost.

This problem has several realistic scenarios. For example, an agent or robot may have a map of the city (G), knowing that some of the roads can be blocked. The agent can perform some actions to get that information, such as calling the city info center, incurring a cost. This scenario is relevant for GPS navigation, as the GPS navigator has a map of the city and knows the location of the car but information of currently blocked streets may not be available in the GPS system itself.

2.1 Related work

Since even the basic CTP is computationally intractable, researchers dealing with this problem achieve positive theoretical results by adding restricting assumptions, such as graphs consisting only of disjoint paths *and* i.i.d. distributions of weights and blockages [Nikolova and Karger, 2008] or requiring that blockages only last a short time [Papadimitriou and Yannakakis, 1991; Bar-Noy and Schieber, 1991].

Partially Observable Markov Decision Processes (POMDP) offer a model for cost optimization under uncertainty. A variant of POMDP proposed by Hansen [Hansen, 2007], called indefinite horizon POMDPs, is particularly relevant here. POMDPs use the notion of a *belief state* to indicate knowledge about the state. In CTP the belief state is the location of the agent coupled with the belief status of the edges (the hidden variables). One could specify edge variables as ternary variables with domain: $\{$*traversable, blocked, unknown*$\}$, thus the belief space size is $|V|3^{|E|}$ (where is the agent, what do we know about the status of each edge). An optimal policy for this POMDP, is a mapping from the belief space into the set of actions (both sensing and moving), that minimizes the expected total cost.

Solving POMDPs is PSPACE-hard (in the number of belief states) in the general case. Although some approximation techniques, such as point-based approximations, show promise (e.g., [Shani *et al.*, 2006]), the size of the state-space of our problem is beyond the current state of the art of POMDP approximation, since the belief state-space size itself is exponential in the size of the graph.

An alternate description of the optimal policy, as a policy tree, may also result in a tree that has size exponential in the number of hidden variables. The time it takes to compute it may be orders of magnitude larger. Thus, such policies can be computed and described only for very small input graphs. For example, the PAO* algorithm [Ferguson *et al.*, 2004] optimally solves basic CTP, but was reported to apply only for a very small number of hidden variables (10 "pinch points" in the experimental results supplied in that paper).

Another related scheme is the PPCP algorithm [Likhachev and Stentz, 2006] which was applied to navigation on uncertain terrain. PPCP uses reverse A* to generate a policy based on *clear preferences*. In succeeding iterations, the policy is improved, until convergence, with optimality guarantees under certain assumptions. However, there is a clear assumption in PPCP that every action may reveal only one hidden variable. In CTP, when arriving at a node, the status of all its incident edges are revealed. Thus, PPCP is not directly applicable to CTP, and its adaptation to this problem is non-trivial.

3 Theoretical Results

We examine special cases where an optimal policy can be computed efficiently, specifically, the case of disjoint paths, as shown in figure 1.b. [Nikolova and Karger, 2008] also studied CTP with disjoint paths but with completely different assumptions on the distribution and possible costs of edges.

3.1 CTP on disjoint paths without sensing

In order to have a finite expected cost (since it is possible for all paths to the goal to be blocked), we assume a finite (but very large) cost edge between the source and target, that has probability 1 of being traversable as in figure 1.c.

First, consider just *committing* policies: a policy is *committing* if, once traversal along any path i begins (we call this "trying path i") it commits the agent to continue along path i until either the target t or a blocked edge is reached. In the latter case, the agent backtracks to s and chooses another path. Even considering just committing policies is still non-trivial, as for n disjoint paths, there are $n!$ such policies. Let $(e_{i1}, e_{i2}, ...e_{ik_i})$ be the edges composing the ith path. Define $W_{i,j} = \sum_{m=1}^{j} w(e_{im})$, the cost of the first j edges on path i. We define a "backtracking cost" on path i, denoted BC_i, the cost to travel to the first blocked edge and back to s, or 0 if the path is traversable. The expectation of this cost is:[2]

$$E[BC_i] = 2 \sum_{j=2}^{k_i - 1} W_{i,(j-1)} p(e_{ij}) \prod_{m=1}^{j-1} (1 - p(e_{im})) \quad (1)$$

Lemma 1 *The optimal committing travel policy is to try the paths in non-decreasing order of the value D_i, where:*

$$D_i = \frac{E[BC_i]}{Prob(\textit{path i is traversable})} + W_{i,k_i} \quad (2)$$

[2] **Explanation:** We sum over every possible j where edge e_{ij} is the first blocked edge, the cost of getting to edge e_{ij} and back. Each case is multiplied by the probability that all preceding edges are traversable, and e_{ij} is blocked.

Proof (Outline): For disjoint paths, a committing policy is equivalent to an ordering of the paths to be tried. The rest is proved by contradiction: assume the existence of an optimal ordering of the paths where the non-decreasing property is violated, specifically let $D_i > D_{i+1}$. Simply switching D_i, D_{i+1} creates a policy that can be shown to be better, a contradiction. □

Lemma 2 *For disjoint paths, there exists a committing policy that is optimal among* all *policies.*

Proof (outline): Since it is clearly suboptimal to traverse edges that have been traversed before without reaching an unknown edge, we can re-formulate CTP over disjoint paths by defining a new action space consisting of two types of macro actions, all starting from s:

1: TRY$_i$: attempt to directly reach the goal through path i, returning to s if a blocked edge is encountered. This macro action can have outcomes: "success" (placing the agent at t), or "blocked at j", placing the agent at s.

2: INSPECT$_{i,j}$: attempt to reach **up to** the jth edge in path i, and return to s regardless of whether that edge is blocked. This macro action can have outcomes: "unblocked" (until j) or "blocked at l" (where $l \leq j$), both placing the agent at s.

Consider an optimal policy π with one or more $INSPECT$ actions. The last macro action in every path in π must be a TRY action, for π to be a complete policy that reaches the goal. Thus, there exists a subtree T of π rooted at a node v, labeled by some $INSPECT_{i,j}$ action, such that all subtrees T' of T have only TRY actions. Observe that each sub-tree T' is actually a committing policy, which can be fully described by a total ordering over the TRY actions in the subtree. We can now show that π can be improved by replacing the $INSPECT_{i,j}$ action at v by a TRY_i action, which contradicts the optimality of π. In order to do so, we make use of Lemma 1, and the fact that $INSPECT_{i,j}$ never increases D_i when its outcome is "unblocked". □

Immediately following from Lemmas 1 and 2, we have:

Theorem 1 *The optimal travel policy for disjoint paths is to TRY the paths in order of non-decreasing D_i.*

3.2 CTP with sensing on disjoint paths

Adding remote sensing actions makes computing the optimal policy hard even for disjoint paths. However, optimality can be guaranteed for special cases with the following condition — a given set S of sensing actions is currently necessary but once we perform one "successful" sensing action from S (where *success* depends on context, as discussed below) no further sensing from S is required. Examples for such cases are presented below. Let $p(s_i)$ be the probability a sensing action on e_i succeeds. In such special cases, we can show:

Lemma 3 *Ordering the sensing action in non-increasing order of $\frac{p(s_i)}{SC(e_i)}$ is an optimal sensing policy.*

Proof (outline): The sensing policy here is a permutation O over the sensing actions, with expected sensing cost:

$$E[SC(O)] = \sum_{i=1}^{n} SC(e_{O(i)}) \prod_{j=1}^{i-1} (1 - p(s_{O(j)})) \qquad (3)$$

Let O' be an optimal ordering that violates the lemma. Then there must be two consecutive sensing actions that are in increasing order in O'. It is easy to show in the above summation that switching them in O' results in a smaller expected cost, a contradiction. □

In general, it is very hard to determine whether the conditions of Lemma 3 hold. But it is applicable in the simple graph shown in figure 1.c. There are n paths to the goal, each with two edges. In each path, the first edge is always traversable at a cost of 1, and the second edge e_i has zero cost, but is blocked with probability $p(e_i)$. A additional traversable "default" edge with a very large cost W that goes directly to the goal also exists. The robot can sense (from the initial location) the status of an edge by paying a cost $SC(e_i)$ or by physically traveling to an incident vertex. In this case, we say that a sensing operation is "successful" if it discovers that an edge is unblocked (it can now follow that path to the goal and no further sensing is required). Thus $p(s_i) = 1 - p(e_i)$. Assume that $SC(e_i) > 1$, and that $2p(e_i) > SC(e_i)$, for all $1 \leq i \leq n$. In this case, algorithm 1 below applies:

Algorithm 1 Optimal policy for special case

```
while there is an unsensed edge
    Perform sensing of an edge with highest (1−p(e_i))/SC(e_i).
    If e_i is unblocked, travel through it to the goal.
endwhile
Use edge e_{n+1} to travel to the goal.
```

Theorem 2 *: Algorithm 1 reaches the goal with minimal expected cost.*

Proof (outline): The costs are such that it never pays to try to move the robot to a vertex unless it is certain that the next edge is traversable. After the first successful sensing action a traversable path is found and no more sensing actions are needed. Thus, the problem is equivalent to finding the permutation O of $[1 : n]$ that minimizes the expected sensing cost (of equation 3). This is achieved by having $\frac{1-p_{O(i)}}{SC(e_{O(i)})}$ monotonically non-increasing, due to Lemma 3. □

Observe that allowing dependencies between edges in this simple graph, we can prove that finding the optimal plan with dependencies is NP-hard. Proof is by reduction from minimum DNF cover [Garey and Johnson, 1979].

Another scenario where lemma 3 is applicable is for a disjoint path, when all sensing operations must be done before moving, and only along that path. When considering a set of sensing actions, Lemma 3 allows us to find the optimal ordering, except that in this case a sensing operation is considered "successful" if it detects that the respective edge is *blocked* (and further sensing is not needed). Thus $p(s_i) = p(e_i)$. The following result follows immediately from Lemma 3.

Theorem 3 *For a set of sensing operations to be made along a single disjoint path, the optimal sensing policy is to perform the sensing operations in non-increasing order of $\frac{p(e_i)}{SC(e_i)}$.*

This theorem can be also applied as a heuristic in some cases of non-disjoint paths, as the indicated policy is "locally" optimal for this sequence of observations, (in the sense that it ignores potential benefit of the gathered information w.r.t. future travel that may use the same edges). For example, the

Algorithm 2 Pseudo-code for SN

```
procedure main (Graph G, vertex s, vertex t)
{01}    OG = G;
{02}    v = s;
{03}    Update OG based on local sensing;
{04}    while(v ≠ t) do
{05}       P=CalculatePath(OG,v,t);
{06}       v=TraversePath(P);
{07}    endwhile

vertex function TraversePath (Path P)
{08}    While P ≠ ∅ do
{09}       (x, y)=pop first edge from P.
{10}       if((x, y) is blocked)
{11}          return(x);
{12}       if (VerifyPath(P)==FALSE)
{13}          return(x);
{14}       Traverse(x, y) and place Agent at y;
{15}       Update OG based on local sensing;
{16}    endwhile
{17}    return(t);

bool function VerifyPath (Path P)
{18}    foreach(unknown e ∈ (P ∩ OG))
{19}       if(shouldSense(e))
{20}          if(sense(e)==BLOCKED)
{21}             delete e from OG;
{22}             return(FALSE);
{23}       endif
{24}    endfor
{25}    return(TRUE);
```

"always sense" policy used in our experiments below commits to remote sensing of the entire path it intends to travel. Here, it is "locally" optimal to perform the sensing operations using this scheme, and this is shown empirically to be beneficial even when the indicated path is not disjoint.

4 CTP with sensing in the general case

We now turn to *CTP with sensing* in a general graph. As indicated above, solving this problem optimally is intractable and even representing the optimal policy requires exponential space in the worst case, necessitating a heuristic approach. We follow the common heuristic approach, which removes some of the problem's constraints in order to generate heuristics. One such direction is to use the *free-space assumption* [Koenig *et al.*, 2003]. The assumption is that any part of the "space" is free (in our case an edge is unblocked) unless specifically known otherwise. We propose the *Free-space assumption Sensing-based Navigation* scheme (FSSN) for navigating in physical graphs.

FSSN is a navigation algorithm that plans a path to the goal but allows plugging in different *sensing policies* for deciding which edges to sense during the moves. In FSSN the agent maintains an optimistic graph (OG), initially a copy of G, which contains all edges which are either *known to be traversable* or *unknown* (OG unifies *traversable* and *unknown* in the belief state). Outcomes of sensing (remote or local) on *unknown* edges are updated based on the agent's knowledge: edges found to be *blocked* are deleted from OG.

The pseudo-code for FSSN is presented in Algorithm 2. The agent first plans a path P on OG from v to t with any known path finding algorithm, such as Dijkstra's algorithm or a heuristic search algorithm based on A* (line 5). The agent can then attempt to traverse P, without sensing (line 6). However, since edges in OG may turn out to be blocked, this may result in wasted travel cost when a blocked edge on P is physically reached. Therefore, the agent may decide to interleave sensing actions on P (and pay their costs) into its movements to update OG prior to continuing along the path. The decision on which edges from P to sense is done by the *sensing layer* in the *verify path* which is invoked in line 12. The *ShouldSense*() function (line 19) is activated on all unknown edges of P. *ShouldSense*() is a key function: different implementations determine the actual behavior of FSSN. We describe several such implementations below. If no blocked edges are detected by sensing, the agent moves one step along P (line 14) and the process is repeated. If a blocked edge in P is detected (line 13), and thus OG changes, a new path to the goal is calculated and the process is repeated.

The basic simplification in this method is that edges that were not sensed are assumed (according to the free space assumption) to be traversable. Thus, if the shortest path P is not known to be blocked it is assumed traversable and the agent just takes it. The benefit of this approach is that no complicated calculations are needed on expected costs of each action, making the memory for applying the policy linear, rather than exponential, in the size of the graph. In addition, the sensing policies may choose to sense edges that are believed to be essential.

5 Sensing policies

We propose a number of the sensing policies for FSSN. Perhaps the two simplest policies are those that ignore the sensing costs altogether. These approximate many existing approaches to navigation.

5.1 Brute force policies

The brute force *Never Sense* (BFNS) policy **never** senses any remote edges. As edges are sensed automatically with local sensing, no sensing cost is ever incurred by BFNS, but it may lead to increased travel costs when a blocked edge is only recognized by local sensing. In fact, FSSN with BFNS is a free-space assumption algorithm for basic CTP.

In contrast, the brute-force *Always Sense* (BFAS) policy takes the opposite direction. In BFAS, the agent queries **all** remaining edges in the path before it moves through it. If an edge on the path is discovered to be blocked, BFAS recomputes the shortest path to the goal. Only after the entire path is proved to be traversable does the agent perform a sequence of moves along the computed path. The BFAS policy optimally minimizes the travel cost, but may incur a large sensing cost. An agent using BFAS can still try to keep down the sensing cost by trying to *optimize the order* in which sensing operations of unknown edges on the path are attempted, *by using theorem 3*, which we evaluate empirically below.

5.2 Value of Information polices

The sensing layer should focus on determining the merit of new information gathered by each possible edge-sensing operation in order to make sensing decisions. There are four scenarios to consider for each edge. If we choose to sense, the edge might turn out to be traversable (this scenario is labeled S^+) or to be blocked (labeled S^-). Alternatively, if we choose not to sense, the edge may be traversable (NS^+)

or blocked (NS^-) but we do not know this unless we physically get there. We can now define the expected cost of the two different decisions (sense and not sense) as follows:

$$E(sense(s,e)) = (1-p(e))E(S^+) + p(e)E(S^-) \quad (4)$$

$$E(\neg sense(s,e)) = (1-p(e))E(NS^+) + p(e)E(NS^-) \quad (5)$$

The value of information (VOI) of sensing an edge e is defined as $VOI(e) = E(\neg sense(s,e)) - E(sense(s,e))$. Now if the VOI of a given sensing action is greater than the corresponding sensing cost ($(VOI(e) > SC(v,e))$) then it is beneficial to perform the sensing action and the $shouldSense()$ functions recommends this.

Observe that to compute a *true value of information*, one should compute the above expected costs assuming optimal future behavior, which is itself intractable. In order to *practically* approximate the value of information, one makes simplifying (usually myopic) assumptions. One such set of assumptions is that we will only do a single sensing operation before beginning to move, never do another sensing operation, and commit to the sensing operation that shows the highest net value under these assumptions. In CTP with sensing, even under this strict set of assumptions, we must compute the expected cost of an optimal motion policy without sensing. However, since CTP is itself intractable, and the optimal policy may require exponential space we propose the idea of "value of information given a non-optimal subsequent policy" as an approximation. In particular, we examine the case where the subsequent policy is to act under the free-space assumption, but this concept can be generalized to other efficiently computable policies.

Expected Total Cost with Free Space Assumption.

A computationally efficient heuristic for sensing adopts the *free space assumption* yet again for the purpose of estimating the post-sensing path cost. The idea is to assume (for the sake of the decision making about sensing) that every edge in OG is traversable, except for the edge e under consideration.

To demonstrate the dilemma of whether to sense or not consider Figure 1 again. The *expected cost with the free space assumption* (EXP) heuristic uses Equations 4 and 5 to estimate the VOI, except that we use $E(S^+) = E(NS^+) = w(s \rightarrow t)$ for cases where e is traversable, and $E(S^-) = w(d(s))$, $E(NS^-) = w(s \rightarrow v) + w(d(v))$ for the cases where e is blocked. The action with the minimal expected cost is chosen. In our example of figure 1.a, suppose that $SC(s,e) = 1$ and that $p(e) = 0.5$. The expected cost for sensing e from s is $E(sense(s,e)) = 0.5 \times 8 + 0.5 \times 12 = 10$. Likewise, the expected cost of not performing the sensing is $E(\neg sense(s,e)) = 0.5 \times 8 + 0.5 \times 16 = 12$. According to the expected cost policy the agent will choose to perform the sensing because the net VOI for sensing e is greater than 1.

FSSN Single-step VOI

The EXP policy acted under the unrealistic assumption that all edges other than edge e in question are traversable. We wish to estimate the true expected cost of navigation under FSSN. This is called the VOI policy. Observe that this requires a summation of a number of graphs exponential in the number of unobserved edges. As an exact efficient scheme for doing so is not known, we approximate the expected travel cost by sampling. For each of the four scenarios used in Equations 4 and 5 we generate sample graphs where unknown edges are blocked according to their respective $p(e_i)$. For each such graph G_{sm}, we execute the FSSN navigation scheme with no remote sensing. That is, we find a path on OG, try to follow it until we hit a blocked edge, repeatedly, until the target is reached, and record the total path cost for G_{sm}. These costs are averaged across samples, and used in Equations 4 and 5 when estimating the VOI.

6 Experimental results

We have experimented with the above policies, using two sensing cost functions, inspired by real-world settings:

Constant cost: Assumes a constant $SC(v,e) = c$. For example, in a city, a car driver might call up roadside assistance which charges a constant amount per information item. If $c = 0$, the expected cost policies converge to the *always sense* policy, since there is nothing to lose by sensing. Likewise, if $c = \infty$, the expected cost policies converge to the *never sense* policy, since any travel cost is cheaper than a sensing action.

Distance cost: Assumes that cost is proportional to distance from the current location of the agent to the sensed item. Formally, for an edge $e = (x,y)$ we define:
$SC(v,e) = c \cdot \min(dist(v,x), dist(v,y))$.

Our experimental setting aims to simulate graphs that could correspond to a small city, and we used Delaunay graphs [Okabe *et al.*, 1992] which are common for this. First, a Delaunay graph of 50 vertices was created by placing the vertices in a square of size 100×100 units, resulting in 94 edges. Then, two vertices were selected as the start and target vertices. Now, each edge of the graph was blocked with a fixed blocking probability (BP). We ran the different policies on 50 such different cases and report the average cost below.

Table 1 shows the results for different sensing costs. For each cost we tried four BP values of 0.1, 0.3, 0.5 and 0.6. For each case we report the *travel*, *sensing*, and *total* costs for each of the four policies discussed in the paper.[3] In the VOI policy we sampled 500 cases.

In cases with very small sensing costs, the relative cost of sensing is much cheaper than the cost of traveling. Thus, there is almost no reason not to sense a questionable edge. Indeed, the BFAS policy usually resulted in the lowest total cost. Similarly, in the opposite cases, where sensing was very expensive BFNS should be the policy of choice. These obvious cases are not shown in the table.

The first section of table 1 corresponds to a constant sensing cost of 5. With $BP = 0.1$ most of the edges are traversable. Thus, the BFNS policy of not sensing anything proved best. But, with larger values of BP the VOI policy proved best. It is important to note that the EXP policy which is much simpler to compute (a few milliseconds for these graphs), performed comparably well, and even won by a slight margin in the case of $BP = 0.5$.

Table 1 also shows results for distance cost function with coefficient (c) of 0.04 and 0.2. For 0.04 we also show

[3]We also experimented with an optimal policy but for very small graphs. Our heuristics were not significantly worse than optimal but no real conclusions can be drawn from these graphs. We also tried a random policy and it was always worse than all other policies.

Constant sensing cost of 5												
Policy	**travel**	**sense**	**total**	**travel**	**sense**	**total**	**travel**	**sense**	**total**	**travel**	**sense**	**total**
BP	p=0.1			p=0.3			p=0.5			p=0.6		
Never	157.37	0.00	**157.37**	227.04	0.00	227.04	317.51	0.00	317.51	313.24	0.00	313.24
Exp	157.37	0.67	158.04	206.16	20.00	226.16	258.30	37.50	**295.80**	283.25	49.33	332.58
VOI	157.37	1.00	158.37	201.37	15.00	**216.37**	264.47	46.17	310.64	246.27	48.00	**294.27**
Always	147.83	57.17	205.00	165.53	136.17	301.70	188.32	253.50	441.82	191.55	215.83	407.38
Sensing cost of distance * 0.04												
Policy	**travel**	**sense**	**total**	**travel**	**sense**	**total**	**travel**	**sense**	**total**	**travel**	**sense**	**total**
BP	p=0.1			p=0.3			p=0.5			p=0.6		
Never	157.37	0.00	157.37	227.04	0.00	227.04	317.51	0.00	317.51	313.24	0.00	313.24
EXP	154.80	2.17	156.98	203.60	8.57	212.17	275.63	15.20	290.83	264.14	14.10	278.24
VOI	151.15	4.66	**155.81**	202.45	6.36	**208.82**	245.32	21.5	**266.83**	235.76	13.65	**249.41**
$Always_R$	147.83	38.99	186.83	165.53	101.02	266.55	188.32	194.17	382.48	191.55	155.29	346.84
Always	147.83	27.49	175.32	165.53	57.44	222.97	188.32	102.61	290.93	191.55	101.16	292.71
Sensing cost of distance * 0.2												
Policy	**travel**	**sense**	**total**	**travel**	**sense**	**total**	**travel**	**sense**	**total**	**travel**	**sense**	**total**
BP	p=0.1			p=0.3			p=0.5			p=0.6		
Never	157.37	0.00	157.37	227.04	0.00	227.04	317.51	0.00	317.51	313.24	0.00	313.24
EXP	156.21	0.85	**157.06**	208.78	11.55	220.33	271.96	25.59	**297.55**	268.34	28.96	**297.30**
VOI	156.21	0.85	**157.06**	199.13	16.12	**215.26**	266.62	40.21	306.83	250.32	54.35	304.66
Always	147.83	137.44	285.28	165.53	287.20	452.73	188.32	513.05	701.37	191.55	505.80	697.35

Table 1: Results with different sensing costs. The best sensing policy in each group is highlighted in **bold**.

an implementation of BFAS where we randomize the order of the sensing actions (this is labeled $Always_R$). Observe that adopting the policy of Theorem 3 for "always sense" is clearly a winner. In all the possible cases both expected costs policies systematically outperformed the two brute force polices in their total costs and the VOI policy was the best in most cases. In a small number of cases, the EXP policy was best but the VOI policy did not do much worse. We performed a large number of other experiments (e.g. with varied edge probabilities) and similar tendencies were observed.

7 Discussion and future work

We have introduced optimal policies for CTP in a number of special case graphs and presented two heuristic policies. Both policies proved useful across the sensing cost functions we tried. Single-step VOI is more complicated than EXP but works better in most of the scenarios.

Due to the fact that we had to resort to sampling in order to estimate single-step VOI, the resulting estimate may be noisy. Thus, in addition to estimating the VOI, we can also estimate its sampling variance. The estimated sampling variance can be used in several ways: in order to control the number of samples, or to evaluate risk for not performing sensing. Initial results in applying these methods show promise, but a disciplined treatment of this issue remains for future work.

Another future direction is in extending the theoretical results on disjoint graphs so as to get a better heuristic for the general case. This involves trading off policy size for a better approximation to the optimal behavior.

8 Acknowledgements

This work was supported by the Israeli Science Foundation.

References

[Bar-Noy and Schieber, 1991] A. Bar-Noy and B. Schieber. The canadian traveller problem. In *SODA*, pages 261–270, 1991.

[Felner *et al.*, 2004] A. Felner, R. Stern, A. Ben-Yair, S. Kraus, and N. Netanyahu. PHA*: Finding the shortest path with A* in unknown physical environments. *JAIR*, 21:631–679, 2004.

[Ferguson *et al.*, 2004] D. Ferguson, A. Stentz, and S. Thrun. PAO* forplanning with hidden state. In *ICRA*, 2004.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., 1979.

[Hansen, 2007] Eric A. Hansen. Indefinite-horizon POMDPs with action-based termination. In *AAAI*, pages 1237–1242, 2007.

[Koenig and Likhachev, 2002] S. Koenig and M. Likhachev. D* lite. In *AAAI*, pages 476–483, 2002.

[Koenig *et al.*, 2003] S. Koenig, Y. Smirnov, and C. Tovey. Performance bounds for planning in unknown terrain. *Artificial Intelligence Journal*, 147(1-2):253–279, 2003.

[Likhachev and Stentz, 2006] M. Likhachev and A. Stentz. PPCP: Efficient probabilistic planning with clear preferences in partially-known environments. In *AAAI*, 2006.

[Nikolova and Karger, 2008] E. Nikolova and D. R. Karger. Route planning under uncertainty: The canadian traveller problem. In *AAAI*, pages 969–974, 2008.

[Okabe *et al.*, 1992] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations, Concepts, and Applications of Voronoi Diagrams*. Wiley, Chichester, UK, 1992.

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

[Shani *et al.*, 2006] G. Shani, R. I. Brafman, and S. E. Shimony. Prioritizing point-based POMDP solvers. In *ECML*, volume 4212 of *LNCS*, pages 389–400. Springer, 2006.

[Stentz, 1994] A. Stentz. Optimal and efficient path planning for partially-known environments. In *ICRA*, pages 3310–3317, San Diego, CA, May 1994.