Traversing Probabilistic Graphs

Murali Mani, Alex Zelikovsky, Gautam Bhatia, Andrew B. Kahng
Department of Computer Science
University of California, Los Angeles
{mani,alexz,gautam,kahng}@cs.ucla.edu

February 10, 1999

Abstract

The problem of traversing probabilistic graphs has been studied for a long time. This is because most of the graphs that we come across, whether it is a network of roads or a set of network links are probabilistic in nature. A probabilistic graph is one where there is a probability associated with the existence of every edge. We examine the problem of finding the best strategy for reaching a given target vertex from a start vertex in a probabilistic graph.

1 Introduction

A probabilistic graph is one where there is a probability of existence associated with the edges in the graph. Such kind of graphs occur in a network of roads where there is a probability of a road being blocked due to snowfall, accidents etc. Such graphs also occur in networks, where some links can fail and so there is a probability of existence associated with such links. Various formulations on these graphs have been studied for quite some time. The main concerns in these formulations have been whether the graph is connected, and whether two given nodes are connected.

The earliest results in probabilistic graphs date back to the late 1970's when Valiant showed that evaluating the probability that two given nodes in a probabilistic graph are connected is NP hard [9]. Later, Provan and Ball proved that other problems such as evaluating the probability that a probabilistic graph is connected, approximating the probability that a given probabilistic graph is connected, and approximating the probability that two vertices in a probabilistic graph are connected are NP hard [8].

Approximating the probability that a network is connected was shown to be NP hard, but Karger gave a randomized fully polynomial time approximation scheme for the probability that a given network will fail [3, 6]. This was done by enumerating the approximately minimum cuts in the network, and it was proved that the probability that the network will fail can be approximated using this enumeration. The problem of enumerating approximately minimum cuts has been dealt with in [1, 4, 5].

Another objective studied for probabilistic graphs is to determine the most reliable source in a given network or graph. Linear time algorithms exist for this problem when the graph is a tree [10] or a series parallel graph [2].

The problem that has been considered in these formulations is determining or approximating the probability that a probabilistic graph is connected, or finding the most reliable source in a network. Therefore the graphs in these cases required only one parameter associated with each edge, the probability that the edge exists or not. In this paper, we look for the best sequence in which edges should be traversed such that the expected cost for reaching a given target vertex from a start vertex is minimized. Here we require at least one more parameter for each edge, the cost of traversing the edge. This formulation is of particular interest to a traveler who has to reach a destination city, and there is a probability of the roads getting blocked due to snowfall. With the promise for intelligent transportation systems in the future, the problem of finding the best strategy for reaching a destination city from a given start city, where the traffic on the roads can be estimated, seems to gain prominence. This formulation also has application in networks where this would give the optimal strategy for routing packets.

2 Notations and Definitions

We are given a graph $G = (V, E)$ with the following three parameters associated with each edge e_i.

c_i is the cost of checking whether an edge is blocked or not.
$p_i \in [0, 1]$ is the probability that the edge is not blocked.
d_i is the cost of traversing the edge if the edge is nonblocked.

We define the graph such that if an edge e_i is blocked, then the edge cannot be traversed. We are given a start vertex s and a target vertex t, where $s, t \in V$. Also let $|V| = n$ and $|E| = m$.

In this paper, we use the notion of a traveler in a network of roads. This notion follows from the canadian traveler's problem discussed in [7]. In the case where the graph being considered is a set of network links, the traveler will be a packet in the network.

A *strategy* r to reach the vertex t from s is a function $f : P \rightarrow E$, where P is the set of all possible paths that the traveler can traverse, and E is the set of all edges in G. In other words, the strategy gives the next edge $e \in E$ that the traveler must traverse given the path $g \in P$ that he has already traversed. We place a constraint that when a traveler follows a strategy, the traveler must end up at t, if s and t are connected, and he must end up at s if they are not connected.[1] Therefore the traveler will reach t with probability p, and will reach s with probability $(1 - p)$, where p denotes the probability that s is connected to t. The cost of a strategy, denoted by $C(r)$, is defined as the expected cost that the traveler will pay if he adopts r. Let S denote the set of all possible strategies. The *optimal strategy* is the strategy, $opt \in S$ with the minimum cost. For convenience, we denote the cost of the optimal strategy, $C(opt)$ as OPT.

The Optimal Strategy Problem: Given a graph $G = (V, E)$ with three parameters for each edge $e_i \in E$, find the strategy with minimum cost.

3 Other formulations

There are several variations to the above formulation, based on our definitions of the strategy and the graph. We suggest a list of possible variations below.

[1] With this constraint, the traveler will end up at t if there exists a nonblocked $s - t$ path.

1. *Destructive Testing.* Here we place the constraint that the traveler does not visit previously visited vertices. In this case, the probability of the traveler reaching t might be less than p, the probability that s is connected to t.

2. *Multiple Check.* An edge is said to be a *Multiple Check* edge if the edge which is nonblocked at some time can become blocked at a later time and vice versa. If an edge is a multiple check edge, then the traveler will have to check the edge every time he wishes to traverse that edge. In our formulation, we assume that we can represent whether an edge can be blocked by a random variable, and the flipping of the coin to determine whether an edge is blocked is performed before the start of the algorithm. But the result of this flipping is unknown until the edge is checked.
   A traveler will have to check an edge more than once if he does not remember the history (the edges that he has traversed and their state when he visited them). But in our formulation, the traveler remembers the past history throughout. Therefore the traveler need to check an edge only once.

3. *Two parameter.* A graph is said to be a two parameter graph if there are two parameters associated with each edge, the cost of traversing, and the probability that the edge is not blocked. We study the more general three parameter graph where each edge has a third parameter, the cost of checking to determine whether the edge is blocked or not.[2]

4. *Finite block.* An edge is said to be a *Finite block* edge if the edge can be traversed even when the edge is blocked. This requires four parameters for each edge, the fourth parameter being the cost of traversing the edge when the edge is blocked (this is finite here).

There is another interesting variation to our formulation where we define the strategy as a *DFS traversal.* Let $e(x, y)$ be an edge visited when the traveler is at x. In a DFS traversal, if the traveler happens to be at x again, the edge e will not be visited then. Therefore in a DFS traversal, the traveler returns from a vertex v only when the probability of reaching t from v through vertices, all of which have a "DFS index" greater than that of v is 0. For the series parallel graph, we give an algorithm for finding the optimal DFS strategy.

4 Properties of Optimal Strategies

In this section, we discuss some properties of optimal strategies. Let *opt* be an optimal strategy for a graph $G = (V, E)$. Also let $C(opt) = OPT$. We first prove that the number of edges in *opt* is $\Theta(m^2)$. We then show that obtaining an $(1+\epsilon)$ approximation for the optimal strategy is NP hard. Here we show that there exists an ϵ such that obtaining a strategy r, where $C(r) \leq (1+\epsilon)OPT$ is NP hard.

Observation 1 *The number of edges an optimal strategy traverses is at most* $\frac{m^2}{4} + m$.

Proof. Let $G = (V, E)$ be the graph, and m be the number of edges in the graph ($m = |E|$). Let E_{opt} denote the number of edges that an optimal strategy traverses. Let the instance of the graph have k blocked edges and $m - k$ nonblocked edges. The optimal strategy traverses a nonblocked edge at most once between checking two blocked edges. Therefore we obtain

$$E_{opt} \leq (m-k)(k+1) + k \leq \frac{m^2}{4} + m$$

[2]The main feature of a two parameter graph is that the traveler can check for the existence of all the edges incident on a vertex when he visits the vertex for the first time.

□

Remark 1 *There exists an instance of a graph where the number of edges that an optimal strategy traverses is* $\frac{m^2}{4} + m - 1$ *edges.*

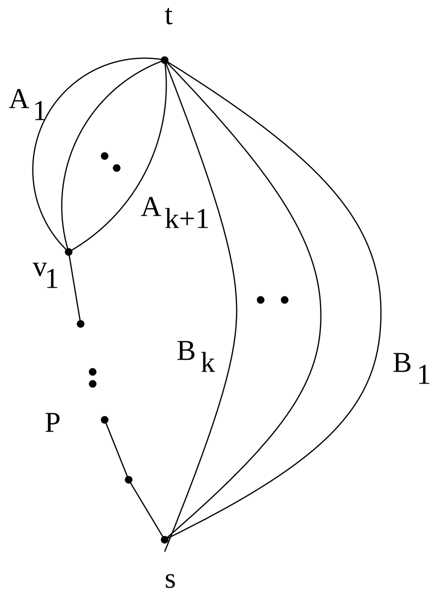

Figure 1: A graph where the number of edges in the strategy can be $\Theta(m^2)$

Figure 1 shows this instance, G. There exists a set of $k+1$ edges from v_1 to t denoted by $A_1, A_2, \ldots, A_{k+1}$. For any edge A_i, the cost of checking associated with that edge is 0, the probability that the edge is not blocked is p and the cost of traversing that edge is a_i. Also there exists a set of k edges from s to t denoted by $B_1, B_2, \ldots, B_k$. For any edge B_i, the cost of checking associated with that edge is 0, the probability that the edge is not blocked is p and the cost of traversing that edge is b_i. Also there exists a path consisting of $2k-1$ edges from s to v_1. We denote this path by P. The cost of checking associated with any edge in P is 0, the probability that the edge is not blocked is 1, and the cost of traversing that edge is 0.[3] Also let $a_i < b_i < a_{i+1}$, for $1 \leq i \leq k$.

We can infer that the optimal strategy for traversing this graph is $P, a_1, P, b_1, P, a_2, P, b_2, \ldots, b_k, P, a_k, P$.[4] The last P in the strategy ensures that the traveler returns to s if he cannot reach t. If the edges A_i, $1 \leq i \leq k+1$ and the edges B_i, $1 \leq i \leq k$ are all blocked, then the traveler will traverse all the edges in the strategy in that order. Let E_{opt} denote the number of edges in the optimal strategy. Then we obtain

$$E_{opt} = (2k-1) + (2k-1)2k + (2k-1) + (2k+1) = 4k^2 + 4k - 1$$

The total number of edges in G denoted by m is given by $m = 4k$. Substituting $k = \frac{m}{4}$, we obtain

$$E_{opt} = \frac{m^2}{4} + m - 1$$

[3] This means that these are dummy edges. But we can always associate a small cost with every edge in P.

[4] This can be inferred from Theorem 2 which is discussed in the next section.

Theorem 1 *There exists an $\epsilon > 0$, such that finding a strategy r where $C(r) \leq (1+\epsilon)OPT$ is NP hard.*

Proof. We prove that the above mentioned problem is NP hard by transforming the connectedness reliability problem which is NP hard [8] to the above problem. We show that if the optimal strategy problem can be solved, then the connectedness reliability problem can be solved for any graph, G. The definitions of both the problems are given below.

$P1$ (The optimal strategy problem)
Given a graph $G = (V, E)$, find a strategy r such that $C(r) \leq (1+\epsilon)OPT$, where $\epsilon > 0$.

$P2$ (The connectedness reliability problem)
Given a graph $G' = (V', E')$, and $s', t' \in V'$, let p denote the probability that s' is connected to t'. The problem is to determine p' such that $p - \delta \leq p' \leq p + \delta$.

In [8], Provan and Ball show that there exists a δ for which $P2$ is NP hard. We transform an instance of $P2$ to $P1$ and show that if $P1$ can be solved in polynomial time then $P2$ can also be solved in polynomial time.

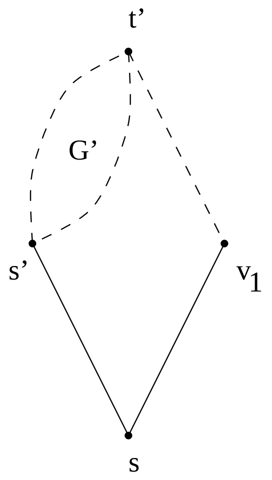

Figure 2: The transformation

Let $G' = (V', E')$ be an arbitrary instance of $P2$. Let the start and end vertices be s' and t'. The problem is to determine the s' to t' connectivity within a δ approximation. Let p_i be the probability of an edge $e'_i \in E'$. We transform G' to an instance of $P1$, $G = (V, E)$ as shown in Figure 2.

G is obtained from G' as follows.
$V = V' \cup \{s, v_1\}$
$E = E' \cup \{(s, s'), (s, v_1), (v_1, t')\}$

In G we wish to find the optimal strategy for reaching t' from s. Let an edge $e \in E$ have three parameters associated with it, $e = \{c, p, d\}$, where c is the cost of checking whether the edge is blocked or not, p is the probability that the edge is not blocked and d is the cost of traversing the edge if the edge is not blocked. The three parameters for the edges in G are as follows.

If $e_i \in E'$, and the probability of existence of e_i is p_i,[5] then the parameters of e_i in G are $\{0, p_i, 0\}$. For the remaining edges, the parameters are given by.
$(s, s') = \{0, 1, 1\}$
$(s, v_1) = \{0, 1, 1\}$
$(v_1, t') = \{0, p_2, 0\}$

Now we shall show how we can solve $P2$ in polynomial time if $P1$ can be solved in polynomial time. We need to consider only two sets of strategies. The first set of strategies is where the first edge traversed is (s, s'), and the second set of strategies is where the first edge traversed is (s, v_1). Note that in both these sets of strategies, when the traveler is at s', he will check all the necessary edges in G' before returning to s if needed.[6]

If (s, s') is the first edge in the optimal strategy for $P1$, then

$$\begin{aligned} 1 + (1-p)(1+1+1(1-p_2)) &\leq (1+\epsilon)(1+(1-p_2)(1+1+1(1-p))) \\ 2p &\geq 2p_2 - \epsilon(1+(1-p_2)(3-p)) \\ p &\geq p_2 - 2\epsilon \end{aligned}$$

If (s, v_1) is the first edge in the optimal strategy for $P1$, then

$$\begin{aligned} 1 + (1-p_2)(1+1+1(1-p)) &\leq (1+\epsilon)(1+(1-p)(1+1+1(1-p_2))) \\ 2p &\leq 2p_2 + \epsilon(1+(1-p)(3-p_2)) \\ p &\leq p_2 + 2\epsilon \end{aligned}$$

Our objective is to obtain two values for p_2, p_{21} and p_{22}, such that $p_{21} \leq p \leq p_{22}$, and $p_{22} - p_{21} \leq 2\delta$.[7] We start with the interval $p_{21} = 0$, $p_{22} = 1$. Our initial choice for p_2 is 0.5. For $p_2 = 0.5$, if the first edge in the optimal strategy is (s, s'), then in the next step we choose $p_{21} = 0.5$, $p_{22} = 1$ and $p_2 = 0.75$. Instead, if the first edge in the optimal strategy is (s, v_1), then in the next step we choose $p_{21} = 0$, $p_{22} = 0.5$ and $p_2 = 0.25$.[8] We repeat till we obtain p' with the desired accuracy. The number of iterations required for this is $O(\log \frac{1}{\delta})$. Therefore if $P1$ can be solved in polynomial time for $\epsilon = \frac{\delta}{2}$, then $P2$ can also be solved in polynomial time. Therefore there exists an ϵ such that determining a strategy r with $C(r) \leq (1+\epsilon)OPT$ is NP hard.

5 Special Classes of Graphs

In this section we examine how the optimal strategy can be determined for some special classes of graphs. We first examine graphs which we call the 1-leg three parameter graph. For this graph the optimal strategy can be determined in polynomial time. In the next section, we consider series parallel graphs.

5.1 1-leg three parameter graph

Let the graph $G = (V, E)$ be a 1-leg three parameter graph, where $V = \{s, t\}$ and $|E| = m$. In other words the graph consists of only vertices s and t, and m edges between s and t. Let the edges be denoted by $e_1, e_2, \ldots, e_m$. Such a graph is shown in Figure 3. For such a graph, a traveler can postpone the checking of an edge till he will definitely traverse that edge if he finds that the edge is not blocked. We determine an optimal strategy which satisfy the above property.

[5] If $e_i \in E'$, then it has only one parameter associated with it, the probability that the edge is not blocked.
[6] Any other strategy where the traveler returns from s' before checking all the edges in G', will have a higher cost.
[7] Then $p' = (p_{21} + p_{22})/2$ gives the probability that s' is connected to t' within a δ approximation.
[8] We perform binary search till we get p' with the desired accuracy.

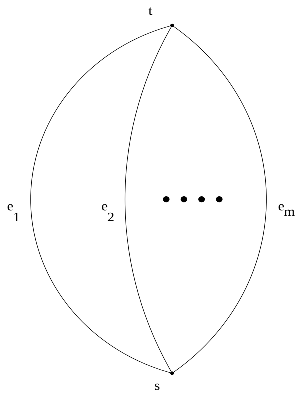

Figure 3: A 1-leg three parameter graph

The objective is to find the least-cost strategy of reaching t from s, which for this graph is to find the order in which the m edges should be checked so that the cost of the strategy is minimized. A strategy can be represented by a concatenation of edges $b_1 b_2 \ldots b_m$, where each b_i represents a distinct edge in E. This representation stands for the strategy where the edge b_i is checked before the edge b_j, if $i < j$. Also let $C(r)$ denote the cost of strategy r. The strategy will be to check the edges till we check an edge which cannot be blocked (the edge always exists), or till all the edges are checked. This is a necessary condition if the traveler must reach t when there exists a nonblocked $s-t$ path. Also as there are only two vertices in the graph, if the traveler cannot reach t, he will end up at s, as required by the definition of the strategy.

The optimal strategy for a 1-leg three parameter graph is determined as follows. Let $e_l \in E$ denote an edge with parameters $\{c_l, p_l, d_l\}$. We define $f(e_l)$ as $\frac{c_l}{p_l} + d_l$. We prove that if $f(e_i) < f(e_j)$ for two edges, $e_i, e_j \in E$, then for any set of paths, X and Y, cost of the strategy $C(Xe_ie_jY) < C(Xe_je_iY)$.

Lemma 1 *If $f(e_i) < f(e_j)$, then for any set of paths, X and Y, $C(Xe_ie_jY) < C(Xe_je_iY)$.*

Proof. The cost of the strategy Xe_ie_jY is the expected cost which the traveler will pay if he first visits the set of edges in X, then he visits e_i, then e_j and finally the set of edges in Y. Let X_v represent the expected cost the traveler will pay when he visits all the edges in X, and let p denote the probability that the set of edges in X will take the traveler to t. Also let Y_v denote the expected cost that the traveler will pay when he visits all the edges in Y. Then $C(Xe_ie_jY)$ is given by

$$\begin{aligned} C(Xe_ie_jY) &= X_v + (1-p) \times (c_i + p_i d_i + (1-p_i)c_j + (1-p_i)p_j d_j + (1-p_i)(1-p_j)Y_p) \\ &= X_v + (1-p)(c_i + p_i d_i) + (1-p)(1-p_i)(c_j + p_j d_j) + (1-p)(1-p_i)(1-p_j)Y_p \end{aligned}$$

Similarly the cost of the strategy Xe_je_iY is given by

$$C(Xe_je_iY) = X_v + (1-p)(c_j + p_j d_j) + (1-p)(1-p_j)(c_i + p_i d_i) + (1-p)(1-p_j)(1-p_i)Y_p$$

$C(Xe_ie_jY) < C(Xe_je_iY)$ *iff* $c_i p_j + d_i p_i p_j < c_j p_i + d_j p_j p_i$. This is true when $f(e_i) < f(e_j)$. Therefore, when $f(e_i) < f(e_j)$, then $C(Xe_ie_jY) < C(Xe_je_iY)$. □

Theorem 2 *The optimal next edge for a 1-leg three parameter graph is given by the one with minimum $f(e)$, among the edges that have not been visited.*

Proof. We prove this by contradiction. Let the optimal strategy, *opt* be $b_1 b_2 \ldots b_m$. Let b_i and b_j be such that $f(b_i) > f(b_j)$ and b_i is visited before b_j. Therefore in *opt*, there exist two consecutive edges between b_i and b_j, say b_k and b_{k+1} such that b_k occurs before b_{k+1}, and $f(b_k) > f(b_{k+1})$. But from Lemma 1, we know that if $f(b_k) > f(b_{k+1})$, then $C(Xb_{k+1}b_kY) < C(Xb_kb_{k+1}Y)$. Therefore, $Xb_kb_{k+1}Y$ is not the optimal strategy. □

For a 1-leg three parameter graph, the algorithm for computing the optimal strategy sorts the edges according to the key $\frac{c}{p} + d$ and keeps them in a list L. The optimal strategy checks the edges in L in that order till the traveler reaches t or all the edges in L are checked.

The algorithm for computing the optimal strategy has a time complexity of $O(m \log m)$ and the maximum number of edges checked by the optimal strategy is m.

6 Series-Parallel Graphs

In this section we examine series parallel graphs and the strategies that can be obtained for this class of graphs. We conjecture that determining the optimal strategy for a general series parallel graph is NP hard. We examine three subclasses of series-parallel graph, for which the optimal strategy can be determined. They are the multi-leg three parameter graphs, the 1-hop three parameter graphs and the 2-branch two parameter graphs.[9] We then examine how approximate strategies can be given for a general series parallel graph. We show that for multi-leg three parameter graphs we can determine the optimal next edge in polynomial time. But for the general class of series parallel graphs, we conjecture that determining the optimal strategy is NP hard. We define the approximation ratio of a strategy as the ratio of the cost of the strategy to the cost of the optimal strategy.

First we shall examine how a set of edges in series or parallel can be replaced by a single edge which is a key substitution for series-parallel graphs.

6.1 Substituting edges in series

Consider two edges e_1 and e_2 in series and with parameters $\{c_1, p_1, d_1\}$ and $\{c_2, p_2, d_2\}$ respectively. The edge e_2 is after e_1 as is shown in Figure 4. The objective is to replace them by a single edge $e = \{c, p, d\}$

Claim 1 $p = p_1 p_2$.

Proof. This is because the probability of reaching t from s must be the same in both the cases. Therefore probability that the edge e is not blocked is the probability that both e_1 and e_2 are not blocked. Therefore we obtain
$p = p_1 p_2$ □

Claim 2 $d = d_1 + d_2$.

Proof. The cost for reaching t from s, and returning to s if no path is found must be the same in both the cases. Therefore we obtain

$$c + pd = c_1 + p_1 \times (d_1 + c_2 + p_2 d_2 + (1 - p_2) d_1) \tag{1}$$

[9] Note that the 1-leg three parameter graph is also a subclass of series parallel graphs.

Now consider that a third edge e_3 is in series with the edges e_1 and e_2, and t is now after e_3. The cost for reaching t from s, and returning to s if there is no path found is

$$c_1 + p_1 \times (d_1 + c_2 + p_2(d_2 + c_3 + p_3 d_3 + (1 - p_3)(d_2 + d_1)) + (1 - p_2)d_1)$$

If we replace e_1 and e_2 by a single edge e, the cost is

$$c + p \times (d + c_3 + p_3 d_3 + (1 - p_3)d)$$

These two costs must be the same. Therefore

$$c + p \times (d + c_3 + p_3 d_3 + (1 - p_3)d) =$$

$$c_1 + p_1 \times (d_1 + c_2 + p_2 \times (d_2 + c_3 + p_3 d_3 + (1 - p_3)(d_2 + d_1)) + (1 - p_2)d_1) \quad (2)$$

From Claim 1 and Equations 1 and 2, we obtain
$d = d_1 + d_2$ □

Claim 3 *$c = c_1 + p_1 c_2 + 2p_1(1 - p_2)d_1$.*

Proof. Substituting Claims 1 and 2 in Equation 1, we obtain
$c = c_1 + p_1 c_2 + 2p_1(1 - p_2)d_1$ □

Theorem 3 *Any set of two edges, e_1 and e_2 with parameters $\{c_1, p_1, d_1\}$ and $\{c_2, p_2, d_2\}$ respectively, can be replaced by a single edge e with parameters $\{c, p, d\}$.*

Proof. From Claims 1, 2 and 3, we know that the substitution maintains the probability and the expected cost. Also any number of edges can be added in series or parallel to them. Therefore the theorem is proved. □

Observation 2 *Any set of m three parameter edges, $e_1, e_2, \ldots, e_m$ in series can be replaced by a single edge, $e = \{c, p, d\}$, where*
$p = p_1 p_2 \ldots p_m$.
$c = c_1 + p_1 c_2 + p_1 p_2 c_3 + \ldots + p_1 p_2 \ldots p_{m-1} c_m + 2p_1(1-p_2)d_1 + 2p_1 p_2(1-p_3)d_2 + \ldots + 2p_1 p_2 \ldots p_{m-1}(1-p_m)d_{m-1}$
$d = d_1 + d_2 + \ldots + d_m$

6.2 Substituting edges in parallel

Consider two edges e_1 and e_2 in parallel and with parameters $\{c_1, p_1, d_1\}$ and $\{c_2, p_2, d_2\}$ respectively as shown in Figure 4. Let the optimal strategy traverse the edges in the order e_1, e_2.[10] The objective is to replace them by a single edge $e = \{c, p, d\}$ which is equivalent to the parallel combination of these two edges in a DFS.[11]

Claim 4 *$p = p_1 + p_2 - p_1 p_2$.*

[10] This order can be determined for any set of parallel edges in a series-parallel graph as is shown later.

[11] In other words, the edges will be checked in the following order. First e_1 is checked and if e_1 is blocked, then e_2 is immediately checked.

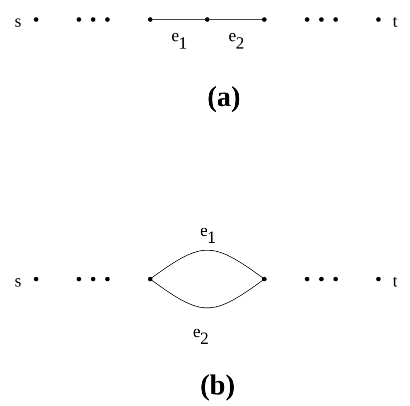

Figure 4: (a) Substituting edges in series (b) Substituting edges in parallel

Proof. The probability of reaching t from s must be the same in both the cases. Therefore probability that the edge e is blocked is the probability that both e_1 and e_2 are blocked. Therefore we obtain
$p = p_1 + p_2 - p_1 p_2$ □

Claim 5 $d = \frac{p_1}{p} d_1 + (1 - \frac{p_1}{p}) d_2$.

Proof. Equating the cost for reaching t from s, and returning to s if there is no path to t, we obtain

$$c + pd = c_1 + p_1 d_1 + (1 - p_1)(c_2 + p_2 d_2) \tag{3}$$

Now consider that a third edge e_3 is in series with the parallel combination of edges e_1 and e_2, and t is now after e_3. The cost for reaching t from s, and returning to s if no path is found is

$$c_1 + p_1 \times (d_1 + c_3 + p_3 d_3 + (1 - p_3) d_1) + (1 - p_1) \times (c_2 + p_2 \times (d_2 + c_3 + p_3 d_3 + (1 - p_3) d_2))$$

If we replace e_1 and e_2 by a single edge e, the cost is

$$c + p \times (d + c_3 + p_3 d_3 + (1 - p_3) d)$$

These two costs must be the same. Therefore

$$c + p \times (d + c_3 + p_3 d_3 + (1 - p_3) d) =$$
$$c_1 + p_1 \times (d_1 + c_3 + p_3 d_3 + (1 - p_3) d_1) + (1 - p_1) \times (c_2 + p_2 \times (d_2 + c_3 + p_3 d_3 + (1 - p_3) d_2)) \tag{4}$$

From Claim 5, and Equations 3 and 4, we obtain
$d = \frac{p_1}{p} d_1 + (1 - \frac{p_1}{p}) d_2$ □

Claim 6 $c = c_1 + (1 - p_1) c_2$.

Proof. Substituting Claims 4 and 5 in Equation 3, we obtain
$c = c_1 + (1 - p_1) c_2$ □

Theorem 4 *In a DFS traversal, any set of two edges, e_1 and e_2 in parallel with parameters $\{c_1, p_1, d_1\}$ and $\{c_2, p_2, d_2\}$ respectively can be replaced by a single edge e with parameters $\{c, p, d\}$.*

Proof. From Claims 4, 5 and 6, we know that the substitution maintains the probability and the expected cost. Also any number of edges can be added in series or parallel to them. Therefore the theorem is proved. □

Observation 3 *In a DFS traversal, any parallel combination of m edges can be replaced by a single edge $e = (c, p, d)$, where*
$p = 1 - (1 - p_1)(1 - p_2)\dots(1 - p_m)$
$c = c_1 + (1 - p_1) \times (c_2 + (1 - p_2) \times (c_3 + \dots + (1 - p_m)c_m)\dots))$
$d = \frac{p_1}{p}d_1 + \frac{(1-p_1)p_2}{p}d_2 + \dots \frac{(1-p_1)(1-p_2)\dots(1-p_{m-1})p_m}{p}d_m$

6.3 Multi-leg three parameter graph

In this subsection we determine the optimal strategy for traversing a multi-leg three parameter graph, where there exists a set of series of edges (paths) between s and t. Such a graph is shown in Figure 5. Let the paths between s and t be $p_1, p_2, \dots, p_k$. The next edge that the traveler will traverse in the optimal strategy can be given by the sequence in which the paths will be traversed. This follows from the following lemma.

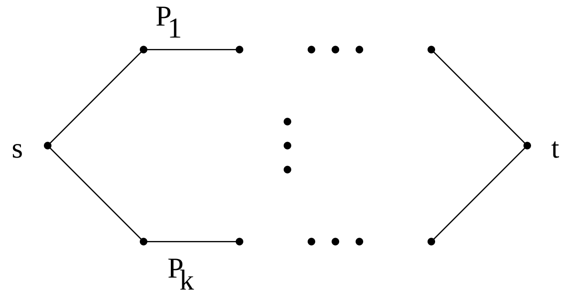

Figure 5: Multi-Leg three parameter graph

Lemma 2 *The traveler will continue on a path till he reaches the last vertex of the path or he finds an edge which is blocked, on which he will return to s.*

The main idea behind the lemma is that a path need to be traversed only when an attempt is made to reach t along that path. From the above lemma, it is clear that the order in which the traveler should traverse the paths will give the actual sequence of edges that the traveler should check.

The optimal strategy is computed as follows. First each path (series of edges from s to t) is replaced by a single edge as in Theorem 3. These edges are then sorted according to the key $\frac{c}{p} + d$, as mentioned in Theorem 2. The sorted list of paths is maintained in a list, L. The optimal strategy traverses the paths in the order till the traveler reached t or all the paths in L have been checked.

Substituting a set of series of k edges by a single edge requires $O(k)$ time. Therefore the time complexity for substituting all the sets of edges in series is $O(m)$. The time required for sorting the paths is $O(m \log m)$. Therefore the time complexity of finding the optimal strategy is $O(m \log m)$. Also the number of edges traversed by the optimal strategy is $O(m)$.

Lemma 2 implies that any series of edges (path) can be replaced by a single edge in any graph. The optimal strategy in this reduced graph will give the optimal strategy in the original graph. This is because once the traveler starts checking a series of edges, he will check the edges in the path till he finds an edge which is blocked or he reaches the end of the path. But the same is not true for a set of parallel edges, the main reason being that the traveler can check one edge in a set of parallel edges, then he can check some other edge not necessarily belonging to this parallel set of edges and then check the remaining edges in this parallel set of edges. This is the reason why it seems hard to obtain the optimal strategy for a general series parallel graph.

6.4 1-hop three parameter graphs

This class of graphs is shown in Figure 6. The main features of this graph are the following.

- There are two edges (paths) from s, (s, v) and (s, t).
- There exists a set of k parallel edges (paths) $\{e_1, e_2, \ldots, e_k\}$ from v to t.
- All the edges in the graph are three parameter edges.

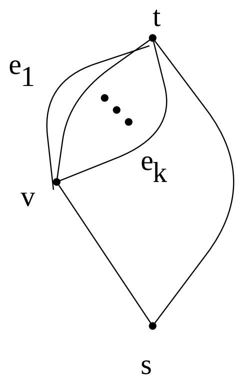

Figure 6: 1-hop three parameter graphs

We call this class of graphs as 1-hop graphs because there exists an optimal strategy where the traveler will visit the edges from a vertex, then return from that vertex and then come back later on to visit the remaining edges from that vertex. But the traveler will come back to a vertex which he returned from *at most once* in the optimal strategy.

Observation 4 *For this class of graphs there exists an optimal strategy which traverses the edge (s, v) at most three times, and all the remaining edges are traversed at most once.*

A brute force algorithm for finding the optimal strategy is as follows.

- Determine the optimal order in which the edges $e_1, e_2, \ldots, e_k$ will be traversed. Without loss of generality let this order be $e_1, e_2, \ldots, e_k$.
- Now consider strategies where the edge (s, t) occurs in between edges e_i and e_{i+1}. Also consider two more strategies, one where the first edge is (s, t) and one where the last edge is (s, t).
- Now the strategy with the least cost among the above $k + 1$ strategies will be an optimal strategy.

The above brute force algorithm requires $O(m \log m)$ time for finding the optimal order among the parallel set of edges (paths). Determining the optimal position of the edge (s,t) requires consideration of $O(m)$ strategies, each of which has requires $O(m)$ time. Therefore the total time complexity of the brute force algorithm is $O(m^2)$.

The brute force algorithm demonstrates the fact that determining the optimal strategy for a 1-leg three parameter graph is sufficient for determining the optimal strategy for this class of graphs in polynomial time. We shall now examine an $O(m \log m)$ algorithm for determining the optimal strategy for the above class of graphs. This algorithm attempts at finding the best strategy where the first edge is (s,v) (let this strategy be denoted by s_1) and compares the cost of this strategy with the cost of the best strategy where the first edge is (s,t) (denoted by s_2). Let the parameters of the edge (s,v) be $\{c,p,d\}$, the parameters of the edge (s,t) be $\{c',p',d'\}$, and the parameters of the edge e_i be $c_i,p_i,d_i\}$. The algorithm is outlined below.

- Determine the optimal order among the parallel set of edges (paths), let it be $e_1, e_2, \ldots, e_k$ (as for a 1-leg three parameter graph). Also find the substitution for this parallel set of edges in DFS. Let the parameters of this "edge" be $\{C_1, P_1, D_1\}$.
- The cost of strategy s_2,denoted by $C(s_2)$, is given by

$$C(s_2) = c' + p'd' + (1-p')(c + p(d + C_1 + P_1 D_1 + (1-P_1)d))$$

- For determining the cost of the strategy s_1 denoted by $C(s_1)$, we distinguish between two cases. One where the last edge checked is (s,t) and the other where the last edge checked is e_k. Let the strategies with the least cost in these two classes of strategies be denoted by s_3 and s_4 respectively. The cost of strategy s_3 is given by

$$C(s_3) = c + p(d + C_1 + P_1 D_1 + (1-P_1)(d + c' + p'd')) + (1-p)(c' + p'd')$$

  To find the cost of the strategy s_4, we first find the optimal order for visiting the edges $\{e_1, e_2, \ldots, e'\}$, where e' is the series substitution for the path $v-t$. The parameters for the edge e' are given by $\{c' + 2(1-p')d, p, d + d'\}$. Let the cost of the optimal strategy among these edges for the 1-leg three parameter graph be C_2. Also let the probability of reaching t with these paths be P_2. Then the cost of the strategy s_4 denoted by $C(s_4)$ is given by

$$C(s_4) = c + p(d + C_2 + (1-P_2)d) + (1-p)(c' + p'd')$$

  After finding $C(s_3)$ and $C(s_4)$, we determine $C(s_1) = \min\{C(s_3), C(s_4)\}$.
- The optimal strategy is the one with minimum cost among s_1 and s_2.

6.5 2-branch two parameter graphs

This class of graphs is shown in Figure 7. For this class of graphs, there is no cost of checking associated with any edge. Therefore the edges in this graph have only two parameters associated with them. As we mentioned before if there is no cost associated with checking edges (two parameter graph), the traveler will check all the edges incident on a vertex when he visits the vertex for the first time. The main features of this graph are as follows.

- There are two edges (paths), one from s to v_1 and the other from s to v_2.

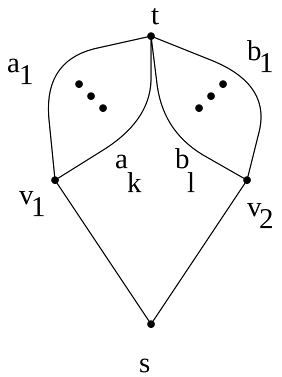

Figure 7: 2-branch two parameter graphs

- Between v_1 and t, there exists a parallel set of edges (paths), $\{a_1, a_2 \ldots, a_k\}$. We shall denote this set of parallel edges by A. Also from v_2 to t, there exists a parallel set of edges (paths), $\{b_1, b_2 \ldots, b_l\}$ which we shall denote by B.

The main characteristic of this graph is that there are only two parameters associated with any edge. For this class of graphs, when the traveler reaches a vertex, he will check for the existence of all the edges from that vertex before he moves from that vertex. The above feature ensures that the when the traveler visits a vertex for the first time, he can determine whether he should return to the vertex again. He also knows the exact cost he will pay when he returns to that vertex. Therefore this class of graphs also has the property that any edge will be visited at most three times, so this class of graphs are also 1 hop graphs.

The algorithm for determining the optimal strategy for this class of graphs is similar to the one used for the 1-hop three parameter graphs. Without loss of generality, let $a_1, a_2, \ldots, a_k$ be the optimal order in which the traveler visits the edges in A, and $b_1, b_2, \ldots, b_l$ be the optimal order in which the traveler visits the edges in B. The set of strategies is divided into two classes, one where the first edge is (s, v_1). We shall denote this set of strategies by S_1. The second set of strategies consists of those where the first edge is (s, v_2). We denote this set of strategies by S_2. Let $s_1 \in S_1$ be the strategy with the least cost among all the strategies in S_1. Also let $s_2 \in S_2$ be the strategy with the least cost among all the strategies in S_2. The algorithm finds s_1 and s_2 and determines the strategy with least cost among the two.

s_1 is determined as follows. We divide the number of possibilities into $k+1$, depending on the first edge in A which is non-blocked. Consider the possibility i that the edge a_i is non-blocked, and the edges $a_1, a_2, \ldots, a_{i-1}$ are all blocked. Now for each i, $1 \leq i \leq k$, we determine the optimal strategy for the remaining graph as in 1-hop three parameter graph. The $k+1$th possibility corresponds to the case when all the edges in A are blocked. After determining the cost of the best strategies for all the $k+1$ possibilities, we find the weighted average of the cost where the weight of possibility 1 is p_1, the weight of possibility $k+1$ is $(1-p_1)(1-p_2)\ldots(1-p_i)$, and the weight of any other possibility i is $(1-p_1)(1-p_2)\ldots(1-p_{i-1})p_i$. This weighted mean gives the cost of the best strategy $s_1 \in S_1$. Similarly the cost of the best strategy $s_2 \in S_2$ is determined. The strategy with the lower cost among these two sets of strategies is the optimal strategy for the graph.

The above algorithm makes $O(m)$ calls to the 1-hop three parameter algorithm. Therefore the total time complexity of this algorithm is $O(m^2 \log m)$.

6.6 Heuristics for series parallel graphs

In this subsection we give two heuristics for obtaining approximate optimal strategies for a series parallel graph. The first heuristic, which we call the "smart DFS strategy" gives the optimal strategy among the set of strategies which are DFS strategies. The second heuristic, which we call "iterative recalculate strategy" provides a better approximate strategy than the optimal DFS strategy. But the approximation ratio (the ratio of the cost of the strategy returned by the heuristic to the cost of the optimal strategy) of both the heuristics is not bounded by a constant.

6.6.1 Smart DFS for a Series Parallel graph

The key idea in determining the smart DFS strategy is the substitution for edges in series and parallel which we discussed earlier. Making this substitution reduces the graph to a 1-leg three parameter graph, for which the optimal strategy can be determined. This need not be the optimal strategy for the series parallel graph because the substitution for parallel edges assumes that the edges are traversed in a DFS order.

The algorithm for obtaining the smart DFS strategy is as follows. The first step is to replace the edges in the graph by a set of 1-leg paths between s and t. Here we replace each s to t path (no two paths have a common edge) by a single edge. This is done by replacing the edges in series and parallel as in Theorems 3 and 4. We perform the replacement of the edges proceeding "backwards" from t to s. For each vertex where there exists a set of parallel edges that take a traveler to t, we also maintain the order in which these parallel edges will be traversed. This order is computed as mentioned in Theorem 5. After computing this order, the set of parallel edges will be replaced by a single edge. After replacing all the s to t paths by 1-leg paths between s and t, the order among these paths is computed as for the 1-leg three parameter graph (Theorem 2). This order is also maintained. The traveler will then traverse the edges in the order till be reaches t or till he finds out that he cannot reach t, in which case he will return to s.

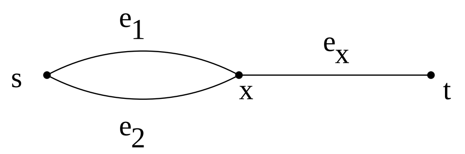

Figure 8: The selection in a smart DFS

Theorem 5 *The next edge which the traveler traverses, when there is more than one edge from the vertex he is currently at, is given by the one with minimum* $\frac{c}{p} + (2 - p_x)d$.

Proof. The cost of reaching t from s when e_1 is traversed first is given by

$$c_1 + p_1 \times (d_1 + c_x + p_x d_x + (1 - p_x)d_1) + (1 - p_1) \times (c_2 + p_2 \times (d_2 + c_x + p_x d_x + (1 - p_x)d_2))$$

This is less than or equal to the cost of reaching t from s when e_2 is traversed first if

$$\frac{c_1}{p_1} + (2 - p_x)d_1 \leq \frac{c_2}{p_2} + (2 - p_x)d_2$$

$\square$

Computing the optimal strategy requires $O(m \log m)$ time. The number of edges that the traveler will traverse in an optimal strategy is $O(m)$.

Remark 2 *The approximation ratio for the smart DFS heuristic in unbounded.*

In other words the ratio of the cost of the strategy returned by the smart DFS heuristic to the cost of the optimal strategy is not bounded by a constant. A graph $G = (V, E)$ where the strategy returned by the smart DFS heuristic can be unbounded is shown in Figure 9. In G, $|V| = 3$ and $|E| = 4$. The three parameters associated with the edge (s, v) (we shall call this edge e_1) are $\{0, 1, 0\}$. The three parameters associated with the edge (s, t) (we call this edge e_4) are $\{0, 1 - \epsilon, N\}$, where $N > 1$ and $0 < \epsilon < 1$. There are two edges from v to t, e_2 and e_3, the parameters associated with them are $\{0, 1 - \epsilon, 1\}$ and $\{0, 1 - \epsilon, \frac{N}{\epsilon}\}$ respectively.[12]

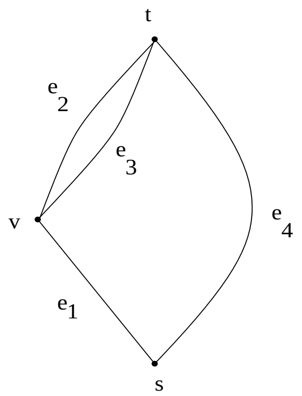

Figure 9: Unbounded Approximation Ratio

For this graph, the smart DFS strategy (which we call *dfs*) is e_4, e_1, e_2, e_3, e_1. The optimal strategy, *opt*, on the other hand is $e_1, e_2, e_1, e_4, e_1, e_3, e_1$. Let DFS denote the cost of the smart DFS strategy, and OPT denote the cost of the optimal strategy. Then we have

$$DFS = (1 - \epsilon)(\epsilon + N + \epsilon N)$$

$$OPT = (1 - \epsilon)(1 + \epsilon N + \epsilon N)$$

$$\frac{DFS}{OPT} = \frac{(\epsilon + N + \epsilon N)}{(1 + \epsilon N + \epsilon N)}$$

Let $N = \frac{1}{\epsilon}$. Now as $\epsilon \to 0$, we obtain $\frac{DFS}{OPT} \to \infty$. Therefore the cost of the smart DFS strategy is unbounded.

6.6.2 Iterative Recalculate Strategy

Here the traveler finds the next edge, say (x, y), for a given graph, $G = (V, E)$ when he is at a vertex x by substituting for series and parallel edges as in the smart DFS strategy. Now the traveler checks this edge (x, y). If (x, y) is nonblocked, the traveler will traverse the edge. Now he will recompute the optimal next edge in G by again substituting the series and parallel edges as is "seen" from his current position y. If (x, y) is blocked, then the traveler recomputes the optimal next edge from v in $G - (x, y)$ by substituting for series and parallel edges.

[12] Note that this graph is a 1-hop three parameter graph, and so the optimal strategy can be determined, but the smart DFS strategy need not be the optimal strategy

Lemma 3 *This heuristic will return a strategy which satisfies Observation 1.*

The above heuristic has the property that between checking two blocked edges, the traveler traverses a nonblocked edge at most once. This ensures that this heuristic does not end up in "cycles", where the traveler will keep moving between two vertices v_1 and v_2 without checking any new edges.

Remark 3 *The approximation ratio for this heuristic is unbounded.*

For the instance shown in Figure 9, the strategy returned by this heuristic will be the same as the strategy returned by the smart DFS heuristic. Therefore the approximation ratio for this heuristic is also unbounded.

7 Conclusions and Future work

In this paper we looked into a new formulation for probabilistic graphs which determines the sequence in which the traveler must visit the edges for reaching t from s. We defined the strategy as the next edge that the traveler should traverse, given the path which the traveler has traversed till now. This formulation is of great importance in Intelligent Transport Systems, where the traffic and road conditions in the future can be estimated, and this information should be exploited in determining the best route for reaching a destination city. We showed that the number of stages in an optimal strategy for any graph is polynomial, but determining the optimal next edge for a general probabilistic graph is NP hard.

We therefore examined the graphs where the optimal next edge can be determined in polynomial time. The graphs we examined are the 1-leg three parameter graphs, multi-leg three parameter graphs, 1-hop three parameter graphs and 2-branch two parameter graphs. All these classes of graphs belong to the more general class of series parallel graphs. The main property of all these graphs is that there exists an optimal strategy such that the number of edges traversed by this strategy is $O(m)$. We then examined the series-parallel graphs, where the number of edges in the optimal strategy can be $O(m^2)$. Determining the optimal next edge seems to be hard for this class of graphs, therefore we imposed the DFS restriction for the strategy and gave an algorithm for determining the optimal DFS strategy. We also gave an iterative recalculate heuristic for determining an approximate strategy which will have lesser cost than the optimal DFS strategy.

Our work on series parallel graphs was based on the conjecture that it is NP hard to determine the optimal strategy for series parallel graphs. Future work could examine formally proving this conjecture. Our attempts were mainly focussed on obtaining an approximate strategy with a bounded approximation ratio for series parallel graphs. The approximation ratio for both the heuristics which we presented are unbounded. Future work could examine whether it is NP hard to obtain an approximate optimal strategy for a series parallel graph, as was shown for a general graph. Our work examined the graphs, where the edges which are nonblocked remain nonblocked, and the blocked edges remain blocked. A more interesting formulation would be one where this need not be true. Future work could examine this possible formulation.

References

[1] C. S. Chekuri, A. V. Goldberg, D. R. Karger, M. S. Levine and C. Stein, "Experimental Study of Minimum Cut Algorithms", Oct 1996

[2] C. J. Colbourn and G. Xue, "A Linear Time Algorithm for Computing the Most Reliable Source in a Series-Parallel Graph with Unreliable Edges", *Theoretical Computer Science*, 1998, pp. 331 - 45

[3] D. R. Karger, "A Randomized Fully Polynomial Time Approximation Scheme for the All Terminal Network Reliability Problem", *Proc. of the 27th Annual ACM Symposium on the Theory of Computing*, 1995, New York, pp. 11 - 17

[4] D. R. Karger and R. Motwani, "An NC algorithm for minimum cuts", *SIAM Journal on Computing*, vol 26, (no. 1), Feb 1997, pp. 255 - 72

[5] D. R. Karger and C. Stein, "A New Approach to the Minimum Cut Problems", Jl of the ACM, vol 43, (no. 4), Jul 1996, pp. 601 - 40

[6] D. R. Karger and R. P. Tai, "Implementing a fully polynomial time approximation scheme for all terminal network reliability", *Proc. of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1997, New York, pp. 334 - 43

[7] C. H. Papadimitriou and M. Yannakakis, "Shortest Paths without a Map", *Theoretical Computer Science*, vol 84, 1991, pp 127 - 50

[8] J. S. Provan and M. O. Ball, "The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected", *SIAM Jl of Computing*, vol 12, (no. 4), Nov 1983

[9] L. G. Valiant, "The Complexity of Enumeration and Reliability Problems", *SIAM Jl of Computing*, vol 8, (no. 3), Aug 1979

[10] G. L. Xue, "Linear Time Algorithm for Computing the Most Reliable Source on an Unreliable Tree Network", *Networks*