Generalized Cost Analysis for Decentralized Search over Metric-embedded Networks

Stephen Macke
Stanford University
smacke@cs.stanford.edu

ABSTRACT

For this project, we consider the problem of routing messages between nodes in Small World [7] and related networks – i.e., networks that are embedded in some metric space, where the probability that a given edge is constructed is inversely proportional to the distance between its corresponding nodes. Our analysis differs from existing approaches in that we assume that each node in the network is assigned a priori some probability of failure independently from some underlying distribution, possibly identically for each node, or possibly from some family indexed by some property of the node (such as its degree). Given a sequence of random $(source, target)$ pairs, we seek to minimize the total expected cost of routing a message from *source* to *target* over a fixed number of runs, where the cost is some function of path length and number of failures. We consider the aforementioned task in the context where each node failure rate is unknown, and each node v is given $\mathcal{O}(deg(v))$ memory with which to estimate the failure rate of each of its neighbors. In this setting, we derive mathematical results about how to route using only local information, and we propose a number of strategies. We further test several of these strategies on synthetic data.

1. INTRODUCTION AND RELATED WORK

The small world phenomenon was first explored empirically by Stanley Milgram through a 1967 experiment in which he examined the number of steps for messages originating from various sources to reach a target in Boston, under the restriction that, at each step, the current holder of the message could only forward it to a first-level acquaintance [5]. Kleinberg was the first to formalize the decision-making process of each message recipient along the way in [3], modeling it as a greedy process. He then considered a family of networks which generalize the model presented by Watts and Strogatz in [7], in which each node in a d-dimensional lattice is connected to its immediate neighbors of L_1 distance 1, as well as q "long-range neighbors", where each long-range edge is selected with probability inversely proportional to some power of the L_1 distance from the corresponding potential neighbor. In this seminal work, Kleinberg showed that, in order for greedy routing to find paths of length $\mathcal{O}(\log n)$ w.h.p., each long-range neighbor v of u must be sampled with probability inversely proprtional to $d(u,v)^\alpha$ with α equal to the lattice dimension d.

Using routing in P2P and other networks as motivation, others have since followed Kleinberg's example and explored aspects of decentralized search. In [4], Manku et al. consider that, for many networks, the GREEDY decentralized search strategy presented by Kleinberg is suboptimal in the sense that the routes discovered are with high probability asymptotically larger than the expected diameter of each network. For example, they consider several classes of network topologies dubbed "Small-World P2P Networks", including the Randomized-Hypercube network, the Randomized-Chord network, and the Symphony network, each of which has diameter $\mathcal{O}(\frac{\log n}{\log\log n})$ w.h.p., but for which the GREEDY strategy takes $\Omega(\log n)$ steps, w.h.p. To remedy this, Manku et al. analyze the "Neighbor-of-Neighbor" (NoN) algorithm, a 2-phase approach which behaves exactly as it sounds. Starting from a given node, it examines all nodes of path length 2 away, then traverses two edges to arrive at whichever node on the ball of radius 2 is closest to the target.

The paper starts by introducing the "Small-World Percolation Graph", which is related to the construction of Kleinberg. For this network, the vertex set lies on a d-dimensional mesh, and the probability that an edge connects two nodes u and v is equal to $d(u,v)^{-d}$ independently of all other edges. (Contrast this to the model of Kleinberg in which each node has a fixed number of long-range edges.) This was shown in [2] to have diameter $\Theta(\frac{\log n}{\log\log n})$ w.h.p. Manku et al. proceed to show that the NoN-GREEDY strategy finds routes of length $\mathcal{O}(\frac{\log n}{\log\log n})$ between any two nodes in this network with probability bounded below by $1 - \frac{1}{n^3}$, where n is the number of nodes.

Next, they consider three "Small-World P2P Networks": the Randomized-Hypercube, the Randomized-Chord, and the Symphony networks. The analysis of Randomized-Hypercube and Randomized-Chord is similar to that of the Small-World Percolation Graph, so that NoN-GREEDY finds paths with lengths of similar bounds, w.h.p. Later, they show that, for these networks, as well as for Small World Percolation Networks, the lower bound for the number of steps of any 1-local algorithm to move between nodes of L_1 distance n away from each other is $\Omega(\log n)$ w.h.p. Here, a 1-local algorithm is loosely any algorithm which uses information only about neighboring nodes when performing search. In particular, GREEDY falls into this category. In addition to these P2P networks, they also consider Skip Graphs [1], for which they prove similar bounds.

Manku et al. then give some experimental results which verify that, indeed, NoN-GREEDY search outperforms GREEDY,

as expected. Lastly, they consider some systems issues for actually implementing NoN-GREEDY search in actual networks, and explore fault tolerance in a couple of differing scenarios. For both scenarios, the underlying assumption is that each node maintains a list of neighbors-of-neighbors. In the first scenario, or the optimistic scenario, nodes are aware of how up-to-date their neighbor-of-neighbor lists are, and perform a GREEDY step in the case of out-of-date lists. This is assumed to occur with probability $\frac{1}{2}$. They also test the pessimistic scenario, in which a neighbor-of-neighbor could be absent, but the message-holder is unaware of this and initiates a NoN-GREEDY hop, only to find out halfway through that the final destination is missing. In this case, it either performs a GREEDY hop, or it restarts the NoN-GREEDY step at the new node. For the optimistic scenario, the algorithm performs similarly to NoN-GREEDY (good). For the pessimistic scenario, the algorithm performs similarly to GREEDY (acceptable) in both strategies considered.

Manku et al. are interested mainly in algorithms and graphs for which it is possible to find paths no larger than the graph diameter w.h.p. As mentioned previously, Kleinberg considered in [3] a family of networks and conditions on these networks under which greedy finds paths bounded by graph diamter w.h.p. However, what if there are conditions one may impose on long-range edges in lattice graphs for which greedy can do even better, w.h.p.? Aspnes et al. consider this precise problem.

Aspnes et al. show lower bounds for greedy routing on a 1-dimensional lattice with very few assumptions in the distribution over which the long-range edges are sampled. For example, when the expected number of links ℓ is in the range $[1, \lg n]$, they show that the expected time to deliver a message between a random pair of nodes is $\Omega(\frac{\ln^2 n}{\ell \ln \ln n})$. Note that this bound is over **all** possible distributions of long-range edges, where the only assumption is that the distribution dependends only on the deltas (distances) away from the node from which the long-range edges originate. This shows that the graphs considered by Kleinberg, in which the probability that a long-range node v is chosen for connection to u is proportional to $d(u,v)^{-1}$ (for the 1-dimensional case), are close to the asymptotic lower-bound for greedy search.

They next consider upper bounds on greedy search for the aforementioned case, where the probability that an edge connects two nodes is inversely proportional to the distance between them. They show some similar (though slightly stronger) results to those of Kleinberg. When the expected number of long-range neighbors is ℓ, where ℓ is in $[1, \lg n]$, it turns out that greedy search takes expected delivery time $\mathcal{O}(\frac{\ln n}{\ell})$.

Next, they consider the previous setting in the presence of node or link failures. If the probability that each node is present is independent and equal to p, the expected delivery time is $\mathcal{O}\left(\frac{\lg^2 n}{p\ell}\right)$, where ℓ is expected out-degree of each node.

In this paper, we focus on routing in the presence of node failures. Kleinberg did not consider this in [3], and the other papers summarized gave this minimal treatment. In the case of Manku et al., the experiments performed only considered the case where nodes were permanently failed, and Aspnes et al. considered, in a very limited 1-dimensional setting, the case where all nodes have the same failure rate. It seems as though this particular nuance should receive more attention. Indeed, in Milgram's original experiment, only about 25% of the messages were eventually received by the Boston target. Furthermore, prior work has always attempted to optimize path length as an objective, which arguably does not model real-world routing very well, where attempting to route to a failed node may incur some additional cost due to, e.g., network timeouts. This paper attempts to give this aspect a more detailed analysis.

The problem we will consider, summarized in section 2, will be that of routing using 1-local information (that is, each node is aware only of its neighbors and their distances from the target, as well as its own internal memory). We provide a rich framework in which to consider cost-minimizing 1-local routing strategies, where "cost" is much more heavily affected by node failures than in prior work.

2. PROBLEM FRAMEWORK

For the task of routing a message from node u to node v in a network using only local knowledge, we define the *cost* of the task as

$$p(u,v) + L(NF(u,v))$$

where $p(u,v)$ is the number of edges in the local-search path taken from u to v, and $L(NF(u,v))$, our node-failure loss function, is some function of the number of failed attempts to transmit the message over one network hop during local search from u to v. For this paper, we will be interested in the case where loss is constant per failed node hop:

$$p(u,v) + L \cdot NF(u,v)$$

Next, let F_v denote the probability that node v is in an unresponsive state. We will consider the scenario where

$$F_{v_1}, F_{v_2}, \ldots, F_{v_n} \overset{iid}{\sim} \Lambda$$

where Λ is some probability distribution with support $[0,1]$.

In this setting, we seek to minimize the expected cost of routing messages between a series of $(source, target)$ pairs. That is, given a set M of messages to route, we seek to minimize

$$\mathbb{E}\left[\sum_{m_{uv} \in M} p(u,v) + L(NF(u,v))\right]$$

Suppose at time t node u attempts to transmit the message to node v, but finds v in an unresponsive state. To simplify

analysis, we will assume that the probability that v is in an unresponsive state at time $t+1$ is independent of whether v was unresponsive at time t.

The networks we consider will be similar to the original formulation of Kleinberg in [3]. That is, we will consider two dimensional meshes where each node u has a fixed expected number of long-range neighbors v, and the nodes v are sampled with probability inversely proportional to the square of $d(u,v)$.

We now consider two possibilities for knowledge granted to the nodes. In the first possibility, we consider the case where, for any node u, u is aware of the values of F_v for each of its neighbors v. We will also consider the setting where u not aware of these values.

For the former case, we note the following observation.

OBSERVATION 1. *In the case where each node is aware of the failure rates of its neighbors, the task of minimizing the expected cost of message delivery reduces to the task of minimizing the message delivery path length in a weighted directed network without node failures.*

To see this, note that, if we force node u to route the message to its neighbor v, the number of attempts to traverse the edge from u to v is a geometric random variable with parameter $1-F_v$, and thus we can assign to $w(u,v)$ the expectated loss of the traversal, namely, $\frac{L}{1-F_v}$. The result then follows for arbitrary paths by applying linearity of expectation. □

As we are interested only in 1-local algorithms, and because there is no compelling reason to assume that greedy performs worse in expectation than any other 1-local algorithm, this case is thus not very interesting, as it is trivial for any node to minimize the expected local cost. We thus consider the more complicated case where each node u is not aware of its neighbors' failure rates. We admit to u $\mathcal{O}(deg(u))$ memory with which to estimate these parameters over a series of runs. The problem can then be modeled as one of exploration versus exploitation, where each node may be likened to a bandit in a multi-armed bandit (MAB) problem.

Note that, because we have a network of bandits rather than a single bandit, any optimality guarantees in the existing literature on the subject do not necessarily apply. For example, in the traditional problem, the number of trials is typically known in advance. In this setting, however, for a fixed number of messages to route, a given node does not in general know how many times it will be traversed.

2.1 Relation to Existing Work

This paper is most similar in spirit to [3], as the network structure is the same, and the search strategies are similarly 1-local. However, in contrast to the existing approaches for decentralized search which give the possibility of node-failures an at-most cursory treatment, this paper puts the issue at center-stage.

3. METHODS

It will be useful to consider the framework previously defined in several restricted settings in order to better perform experimental and theoretical analysis. Here, we consider the case where the node failure probabilities are drawn i.i.d. from a uniform distribution over [0,1] (that is, $\Lambda \sim U(0,1)$). We further assume that the number of messages to route, N, is known in advance, and that always use the same source and target node for each message. We further restrict the nodes so that they always forward to a neighbor closer to the target; that is, the message never backtracks.

Because each node u is allowed $\mathcal{O}(deg(u))$ memory, it can count the number of successful attempts s_v to forward a message to each neighbor v, as well as the number of failed attempts f_v. As each neighbor v fails with unknown probability p_v, there is a natural tradeoff between estimating these parameters while attempting to forward to the next node in such a way as to minimize expected cost.

What if u has never forwarded to v? Clearly it needs some sort of way to start off its estimate. We thus use Laplacian smoothing on the initial counts, so that each node behaves as if it has had 1 successful attempt and 1 failed attempt to forward to each of its neighbors.

In this setting, how should a node choose a neighbor to whom to forward the message? We consider several candidate strategies below. Note that, for each strategy, we keep track of successes and failures, and in the case of a failure at time t, the node chosen by the strategy may change at time $t+1$.

- RANDOM-CLOSER. In this approach, each node forwards the message to a random closer neighbor.
- GREEDY. With this approach, a node u uses its current counts to estimate the expected cost of routing to each neighbor v. It chooses the neighbor with the lowest estimated expected cost.
- EPSILON-FIRST. Inspired by the similar approach for a single bandit, in this setting, the first ε fraction of messages are routed using RANDOM-CLOSER, and the remaining $1-\varepsilon$ fraction of messages are routed according to GREEDY.
- GLOBAL EPSILON-GREEDY. Similar to the single-bandit strategy. A given message is routed using RANDOM-CLOSER with probability ε, and it is routed using GREEDY with probability $1-\varepsilon$.
- LOCAL EPSILON-GREEDY. Each individual node uses an epsilon-greedy algorithm. A given node individually uses RANDOM-CLOSER with probability ε, and GREEDY with probability $1-\varepsilon$, independently of the other nodes through which any message travels. Note that this may be combined with EPSILON-FIRST, so that some proportion of nodes first use LOCAL EPSILON-GREEDY, while the remainder use GREEDY.
- LOCAL-TS, or *local Thompson Sampling*. Drawing upon classic ideas from [6], we assume a prior distribution of Beta$(1,1)$ over each neighbor's failure rate. For each

neighbor v, after we see s_v successes and f_v failures (remember that we initialize these to 1), the posterior distribution over that neighbor's failure rate is $\text{Beta}(f_v, s_v)$. We sample an estimate for v's failure rate p_v^* from this distribution. Using these estimates, we pick the neighbor with the lowest expected cost.

Our goal is then to compare the sorts of performance guarantees offered by each strategy.

4. RESULTS

4.1 Theoretical Results

What does it mean for an algorithm to perform greedily in the framework we have set up? One interpretation is that a greedy algorithm will try to pick a neighbor to send the message in such a way that the expected cost of sending to that neighbor and then routing from that neighbor to the destination node is minimized. If we give the algorithm knowledge of the mean failure rate over all the nodes (without giving it knowledge of the actual failure rates of any of the nodes), we can do precisely this.

Suppose μ is the mean failure rate. (Without seeing any successful or failed attempts to route a packet to a node, this is thus our best estimate.) Given this, as well as the number of long range edges ℓ from each node, the expected cost of routing from a source node to a destination node depends only on the lattice distance d between them. This is because long range edges are added independently at each node, and node failure rates are sampled independently. One fact which makes the analysis particularly nice is that long range edges are sampled uniformly with respect to distance.

Thus, let c_d be the expected cost to route from some node x to a destination of lattice distance d away. With no knowledge of the actual failure rates of any of x's neighbors, any greedy algorithm will attempt to take the edge from x which gets as close as possible to our destination. (Note: if we are considering routing to node x and want to calculate c_d in this manner, it's possible that we actually share some neighbors with x and thus have an idea of the failure rates beyond μ. This is not very likely for small numbers of long range edges ℓ, and we will assume that it never happens for the sake of simplifying the analysis.)

We thus have the following recurrence for the c_k:

$$c_0 = 0$$

$$c_d = \mu \cdot L + \sum_{k=0}^{d-1} w_{dk} c_k$$

where L is our loss for attempting to route to a failed node, and $w_d k$ is the probability that the "best" long range edge from x (i.e., closest to our destination) is of lattice distance k away from the destination.

How do we set the w_{dk}? As an example, $w_{d,d-1}$ is the probability that there are no long range edges from x to any node of lattice distance $[0, 1, \ldots, d-1]$ away from the destination. If our graph lattice diameter is D, we know that the probability a long range node is at distance k away from us is proportional to $\frac{1}{D}$, and for sufficiently large graphs, roughly half of the nodes at distance k from us will also be closer to the destination. Therefore

$$w_{d,d-1} \approx 1 - \frac{d}{2D}$$

Similarly, for $k < d-1$, we have that

$$w_{dk} = \sum_{j=1}^{\ell} \mathbb{P}(j \text{ long range edges to } 0 \ldots d-1) \cdot \mathbb{P}(\text{closest long range edge is at distance } k | j \text{ long range edges})$$

The probability of j long range edges in the range $0 \ldots d-2$ will be approximately

$$\binom{\ell}{j} \left(\frac{d-1}{2D}\right)^j \left(1 - \frac{d-1}{2D}\right)^{\ell - j}$$

and given this, the probability that the closest long range edge to the destination is lattice distance k away is

$$\left(\frac{d-1-k}{d-1}\right)^j - \left(\frac{d-2-k}{d-1}\right)^j$$

since the conditional distribution will follow a discrete uniform distribution supported over integers in $[0 \ldots d-2]$.

If we place the weights w_{dk} in a matrix W, we may write the recurrence as follows:

$$\mathbf{c}_{1\ldots d} = \mathbf{W} \cdot \mathbf{c}_{0\ldots d-1} + \mu \mathbf{L}$$

Here W is a d by d matrix, and $\mu\mathbf{L}$ is the vector with all entries set to μL. If we pad W and $\mu\mathbf{L}$ with zeros appropriately, we get the same unknown $\mathbf{c}_{0\ldots d}$ on both sides:

$$\mathbf{c} = \begin{pmatrix} \mathbf{0}^T & 0 \\ \mathbf{W} & \mathbf{0} \end{pmatrix} \cdot \mathbf{c} + \begin{pmatrix} 0 \\ \mu\mathbf{L} \end{pmatrix}$$

Letting $\mathbf{W}'$ be this padded $\mathbf{W}$ and letting $\mathbf{b} = \begin{pmatrix} 0 \\ \mu\mathbf{L} \end{pmatrix}$, we have that

$$(\mathbf{I} - \mathbf{W}') \cdot \mathbf{c} = \mathbf{b}$$

As the matrix $\mathbf{I} - \mathbf{W}'$ is square and lower triangular, the solution to $\mathbf{c}$ easily obtainable.

We may thus use the above method to precompute the vector $\mathbf{c}$ to use for calculating expected cost from neighbors during routing for all of the algorithms previously mentioned. If the lattice diameter of the graph is D, it requires $\theta(D^2)$ auxiliary space and runs in $\theta(D^2)$ time (assuming $\ell = \mathcal{O}(1)$). Note however that, as an implementation trick, if we delay calculating the weights w_{dk} until we need them, we require only $\theta(D)$ space.

An example use of the vector $\mathbf{c}$ to perform greedy routing is as follows:

- For each of my neighbors v, suppose it is d_v away from the destination.
- The expected cost for v is $L \cdot \frac{failures_v}{failures_v + successes_v} + \mathbf{c}[d_v]$.
- Pick the neighbor v with the lowest expected cost.

Other strategies, like 2-step greedy or Local-TS, use the vector of costs $\mathbf{c}$ in a similar fashion.

4.2 Experimental Results

4.2.1 *Preliminary Results on 10x10 grid*

In what follows, we do not consider Global Epsilon-Greedy to save space. We furthermore do not consider Epsilon-First, and instead note that using a Local Epsilon-Greedy approach for the first ε fraction of messages followed by Greedy for the remainder performed better in practice. This is referred to as "local eps greedy then greedy" in the plots.

We ran simulations on small, randomly-generated 100x100 lattices, where each node was given a single long-range neighbor. We set the cost of attempting to route to a failed node to the value 5, in an attempt to balance against benefits offered by long-range edges. In each case, we routed ≥ 1000 messages from the node in the lower left corner to the node in the upper right corner. The cost of attempting to route to a failed node was fixed at $L = 5$. In most cases, we are interested in routing the first messages with some "training strategy" (e.g. Local Epsilon-Greedy), and the remainder with Greedy. When performing comparisons, we used deep copies of the same network to avoid sharing information about node success / failure counts between strategies.

First, we consider Greedy routing without training, versus Greedy after a Local Epsilon-Greedy training phase of 500 iterations.

Note in figure 1, Local Epsilon-Greedy has a very short spike in cost at the start, then performs slighly better than Greedy. The trained Greedy after 500 iterations then performs much better than the untrained Greedy approach, which is fairly consistently noisy. This is likely because Greedy tends to immediately settle on routing to nodes which have failure rates lower than 0.5, regardless of whether these neighbors have optimal expected cost.

Next, we consider the Thompson-inspired Local-TS approach against the Greedy approach. Note from figure 2

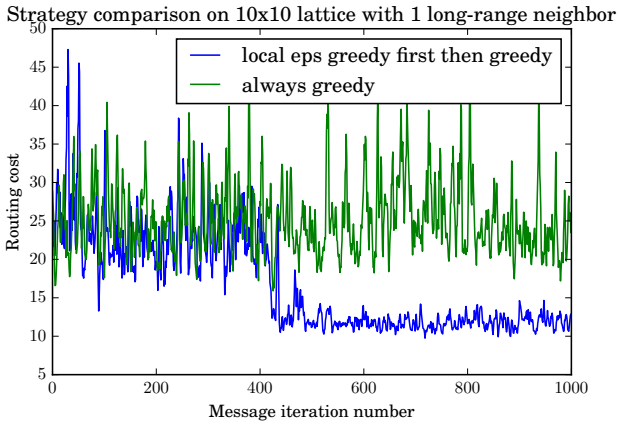

Figure 1: Local Epsilon-Greedy training for 500 iterations, in blue.

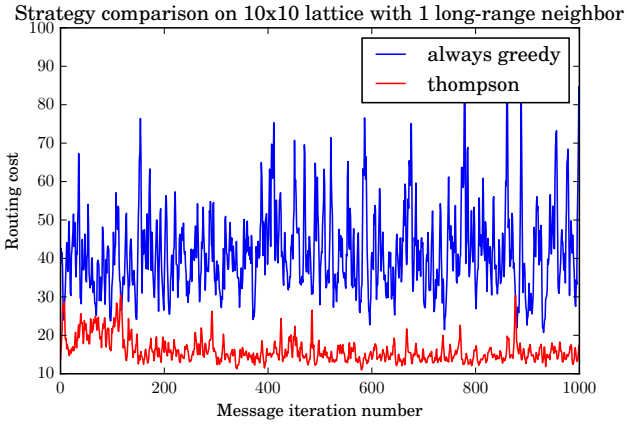

Figure 2: Local-TS outperforms Greedy from very early on.

that Local-TS outperforms Greedy from the start, and has significant self-improvement after about 100 iterations.

We might then ask: how do Local-TS and trained Greedy compare? From figure 3, we see that Local-TS does much better during training and has much less noise, though after training, it is a little difficult to tell how they compare from the current plots.

Thus, let us zoom in. From figure 4, we see that Greedy following some training iterations outperforms Local-TS. Note however, it seems as though the actual time it takes to train Local-TS before it behaves consistently is quite short, suggesting that, due to its inherent randomness, it makes suboptimal choices quite often.

Thus, what happens if we route 500 more messages, but this time allow the Local-TS-trained network to switch over to Greedy? From figure 5, we see that the two approaches perform similarly.

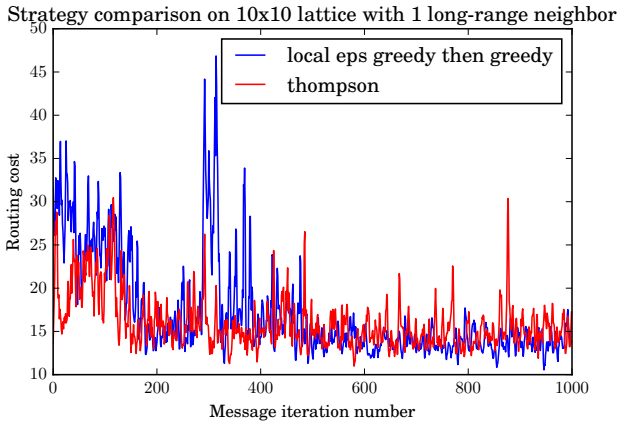

Figure 3: Local-TS performs better during training, but the benefit is lost afterwards.

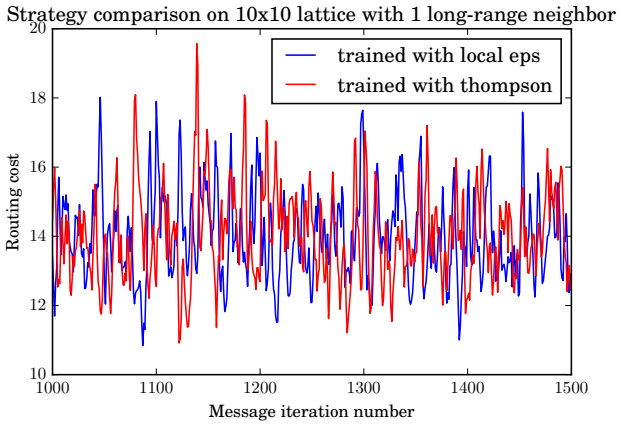

Figure 5: Greedy trained using Local-TS performs similarly to Greedy trained using Local Epsilon-Greedy

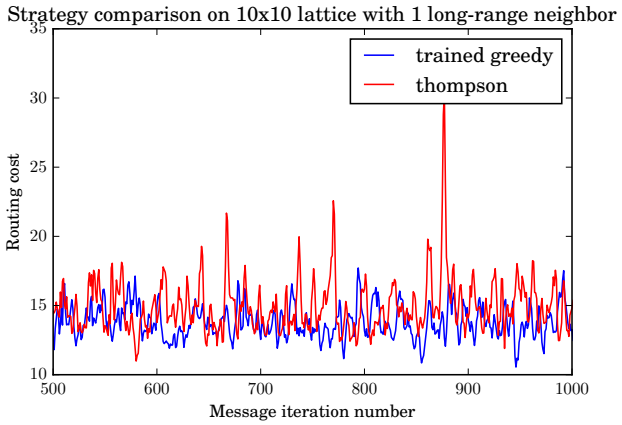

Figure 4: Trained Greedy outperforms Local-TS

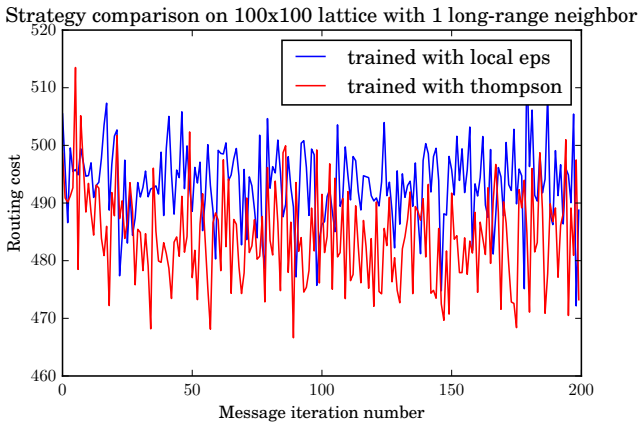

Figure 6: Local-TS outperforms Local Epsilon-Greedy during training, averaged over 2000 runs

4.2.2 Results on 100x100 grid

We now compare strategies under similar parameters, but on a 100 by 100 grid. In all figures except figure 6, the given training strategy was used for the first 100 messages routed, and GREEDY was used thereafter. From figure 6, we see that Thompson sampling on average (over 2000 randomly generated networks) does consistently better than LOCAL EPSILON-GREEDY, as we might have expected from our runs on smaller networks.

How do the various strategies compare as we add more and more long range edges, however? Figures 7, 8, and 9 show that, in all instances, more long range edges per node decrease average (over 100 random graphs) routing cost. However, LOCAL-TS appears to have much better performance over the entire training phase. Indeed, zooming in, we see from figure 10 that this is the case. While other strategies appear to quickly level off during training in the presence of relatively high numbers of long range edges, LOCAL-TS seems to improve consistently.

5. CONLCLUSION AND FUTURE WORK

In this paper, we considered the problem of routing with local information in the presence of node failures. We mathematically generalized the notion of greedy routing in this setting and explored several training strategies. These preliminary results suggest that the Thompson sampling-inspired strategy performs well in practice. A useful direction is to then consider what sort of guarantees can be made about the strategy in this setting.

Further work needs to be done to see how the cost of each strategy scales with network size, as the networks we considered here were consistently small. Further work should also be done to explore how lookahead affects results.

Finally, it could be interesting to slightly extend our nodes to have more knowledge than the base 1-local information we have allowed. For example, in a setting with variable numbers of long-range neighbors, can a node use knowledge

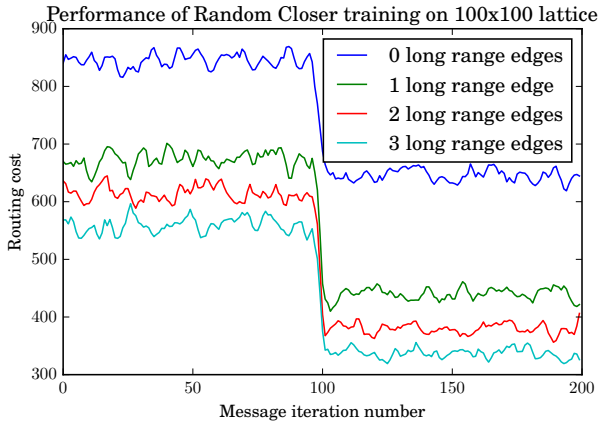

Figure 7: More long range edges improve performance, but lessen the dramatic improvement after the training phase

Figure 8: More long range edges improve performance, but lessen the dramatic improvement after the training phase

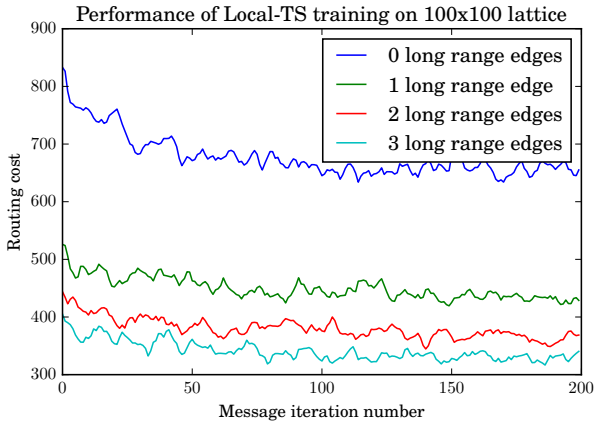

Figure 9: Thompson has better improvement across numbers of long range edges than Local Epsilon Greedy or Random Closer

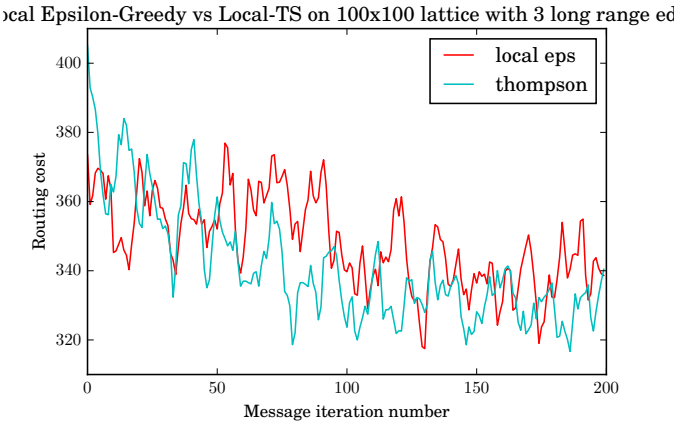

Figure 10: Thompson has better improvement across numbers of long range edges than Local Epsilon Greedy or Random Closer

of the degrees of its neighbors to advantage? Can a node use the iteration number to estimate how often it is being used, and thus switch from explore to exploit if it estimates that messages do not travel through it often? These are optimization problems which become considerably difficult as more complexity is added, but which could potentially lead to better insights about routing in real-world networks.

6. REFERENCES

[1] J. Aspnes and G. Shah. Skip graphs. *ACM Transactions on Algorithms (TALG)*, 3(4):37, 2007.

[2] D. Coppersmith, D. Gamarnik, and M. Sviridenko. The diameter of a long range percolation graph. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 329–337. Society for Industrial and Applied Mathematics, 2002.

[3] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170. ACM, 2000.

[4] G. S. Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 54–63. ACM, 2004.

[5] S. Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[6] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[7] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998.