Abstract

This paper describes a simulated annealing algorithm to compute $k$-connected graphs that minimize a linear combination of graph edge-cost and diameter. Replicas of Internet information services can use graphs with these properties to propagate updates among themselves.

We report the algorithm’s performance on 2- and 3-connected, 50-node graphs, where edge-cost and diameter correspond to physical distance in a plane. For these graphs, the algorithm finds solutions whose edge-cost and diameter are 10% and 70% lower than our graph construction algorithm that returns feasible but unoptimized solutions. When optimizing edge-cost only, the resulting graphs show 25 and 30% reductions edge-cost and diameter, respectively.

1 Introduction

This paper describes a simulated annealing algorithm to construct low diameter, low edge-cost, 2- and 3-connected graphs. We need graphs with these properties to propagate updates between replicas of Internet information systems [2]. For maintaining replicas across the Internet, we use connectivity of 2 or 3 for resiliency, low edge-cost so the physical network is used efficiently, and low diameter to limit transient replica inconsistency.

Since any algorithm that minimizes edge cost or diameter in 2 or greater connected networks is NP-complete [4], we employ simulated annealing [7] to search for approximate solutions. The algorithm presented here simultaneously reduces both edge-cost and diameter by 10 and 70%, respectively, over unoptimized, feasible solutions. With respect to graphs constructed to minimize edge-cost alone, our algorithm finds solutions that show 25 and 30% reduction in edge-cost and diameter, respectively.

1.1 Statement of the Problem

Below, we formally define our topology computation problem, starting with several definitions.

- **$k$-Connected**: A graph $G$ is said to be $k$-connected if no removal of any $k - 1$ vertices together with all their incident edges disconnects $G$.
- **$k$-Connected Regular Graph**: A graph $G$ is said to be a $k$-connected regular graph if all its vertices are exactly $k$-connected.
- **Diameter**: The diameter of a graph $G$ is defined as the maximum shortest path between any two of $G$’s vertices.
- **Degree**: The degree of a vertex $v$ in $G$ is the number of edges of $G$ incident with $v$.

We represent the network topology by graph $G(V, E)$, where the set of vertices $V$ represents network nodes and the set of edges $E$ represent links between nodes. In our simulations, we evaluate complete graphs where the edge costs correspond to physical distance in the plane. Our Internet replication system [1] measures and estimates the end-to-end communication costs between nodes, which depends on network state, physical topology, and processing load at the nodes.

The topology construction problem can be stated as follows. Given the graph $G(V, E)$, a cost matrix $C(E)$, and an integer $k$, construct a subgraph $G'(V, E')$ with the following properties:

- $G'$ is $k$-connected.
- $G'$ has minimum diameter.
- $G'$ has minimum total edge-cost.

In the next section, we review related literature and algorithms for solving similar problems.
1.2 Related Work

Plesnik [8] proves that any algorithm that generates a minimum spanning subgraph of G, say $G'(V,E')$, by selecting $E'$ as subset of $E$ with a given budget constraint and minimum diameter is NP-complete.

Johnson [6] proves that constructing a subgraph which connects all vertices and minimizes the shortest path cost between all vertex pairs, subject to a budget constraint on the sum of its edge-costs, is also NP-complete.

Schumacher [10] provides an algorithm for generating topologies which have minimum number of edges, are $k$-connected and have minimum diameter. However, his method assumes that all the edges have equal weights. We cannot make that assumption since our problem is to build peer-to-peer topologies over real networks.

Steiglitz et al. [11] proposes a heuristic solution to a problem similar to ours. Their problem consists of finding an undirected graph with the following properties:

- Feasibility: The redundancy between any two nodes $i$ and $j$ is at least $R_{i,j}$.
- Optimality: No network which satisfies the first property has lower total edge-cost.

Steiglitz’s algorithm has two parts: the starting and the optimizing routines. The starting routine generates a random feasible solution. The optimizing routine iteratively applies heuristics to generate lower cost topologies. It uses a local transformation called $x$-change, which randomly selects four nodes connected pairwise and swaps the edges connecting them (see Figure 3). It then records the lowest cost feasible topology generated by these local transformations.

Steiglitz’s algorithm to search for low diameter and low edge-cost topologies. We cannot completely map our problem onto the Steiglitz problem. While we can express our connectivity requirement in terms of his redundancy matrix $R_{i,j}$, his optimality constraint cannot express our minimal diameter requirement. This paper extends the Steiglitz algorithm to search for low diameter and low edge-cost, feasible graphs.

Steiglitz’s algorithm has two parts: the starting and the optimizing routines. The starting routine generates a random feasible solution. The optimizing routine iteratively applies heuristics to generate lower cost topologies. It uses a local transformation called $x$-change, which randomly selects four nodes connected pairwise and swaps the edges connecting them (see Figure 3). It then records the lowest cost feasible topology generated by these local transformations.

The algorithm employs hill climbing to generate locally optimal solutions from various starting configurations. After determining a set of feasible solutions, it returns the one with lowest cost.

In [9], Rose uses simulated annealing [7] to find network topologies with small mean distances between nodes, subject to a budget constraint. He shows that annealing help to equalize the initially uneven distribution of mean distances. His algorithm does not guarantee 2- or 3-connectivity. Rose’s local transformation algorithm randomly chooses a node, deletes one of its links and adds a new one to a different node.

1.3 Outline

This paper extends Steiglitz’s and Rose’s algorithms to search for low diameter and low edge-cost topologies. This paper is organized as follows. In Sections 2 and 3 we present our simulated annealing algorithm and evaluate its performance for various graphs and connectivities. Section 4 presents two faster but less optimal heuristics to find low cost, feasible solutions.

2 Our Topology Computation Algorithm

Like Steiglitz’s and Rose’s, our algorithm generates a feasible starting topology and then successively applies one of four transformations to it according to a simulated annealing schedule.

2.1 Generating a Starting Topology

Figure 1 summarizes how we generate a random, feasible starting topology. The goal is to generate a relatively low-cost graph that is likely to be feasible. The algorithm’s inner loop creates a graph where all nodes have degree greater or equal to the connectivity requirement. Of course, this doesn’t guarantee that the graph is $k$-connected or even connected. The outer loop computes the random graph’s connectivity and accepts the graph if it is feasible. The algorithm randomizes itself by relabeling the nodes on every execution. In practice, the fact that there are $N!$ possible starting graphs did not constitute a limitation to the algorithm.

Steiglitz proves [11] that when all nodes have connectivity requirement $k$, the entire graph is feasible if any $k$ nodes meet the connectivity requirement. Therefore, instead of having to check connectivity between all $N^2/2$ pairs of nodes, we only need to check the connectivity of $kN$ pairs of nodes, where $k$ is typically 2 or 3.

To check an individual node’s connectivity we apply the Ford-Fulkerson maximum flow algorithm [3] which computes the number of disjoint paths between a source and a sink in $O(NE^2)$. Figure 2 summarizes how we perform topology feasibility checks.
feasible_start(int N, vector unsatisfied_connectivity, matrix cost)
{
    do
    {
        k = unsatisfied_connectivity(of node 1);
        generate P(i), a permutation on 1..N;
        for all nodes i=1..N, label(i) = P(i);
        do
        {
            A = lowest labeled node with the highest unsatisfied_connectivity;
            unsatisfied_connectivity(A) = unsatisfied_connectivity(A) - 1;
            B = lowest labeled node with the highest remaining
                unsatisfied_connectivity, resolving ties in favor of the
                lowest-labeled node with the cheapest cost to A;
            unsatisfied_connectivity(B) = unsatisfied_connectivity(B) - 1;
            add edge (A, B) to graph;
        } until all nodes have unsatisfied_connectivity = 0;
    } while not check_feasibility(graph, N, k);
}

Figure 1: Generating an initial feasible topology (after Steiglitz).

2.2 Applying Local Transformations

Next, we search for lower cost topologies by repetitively applying four different local transformations to the graph. Steiglitz's used a single transformation, dubbed x-change, that operates on two pairs of connected nodes and their respective edges. X-change deletes the pair of edges between the connected nodes and adds a pair of edges between the nodes that were not formally connected. Figure 3(a) illustrates X-change. Since x-change does not alter node degree, we added three more local transformations that would allow the node degree to increase, so that diameter decreases. We dubbed these three transformations as split, delete, and add.

Split randomly selects a pair of connected nodes, breaks the edge connecting them and connects each of them to another node. Delete randomly chooses a pair of connected nodes with degree greater than the required connectivity and deletes the edge connecting them. Add selects a pair of nodes that are not connected and adds an edge connecting them. Figure 3 illustrates all four transformations.

After each transformation, the resulting topology is checked for feasibility. If the feasibility check fails, the algorithm rejects the resulting topology, and restores the previous one. At each pass through the cost reduction algorithm, we independently select a local transformation to apply. We do not necessarily apply the four transforms with equi-probability. In Section 3, we apply some transformations more frequently than others, depending on whether we want to reduce edge-cost or diameter.

After each local transformation, we check feasibility. In the case of an x-change transformation, this reduces to checking the connectivity of the four, “x-changed” nodes [11]. For a split transformation, this reduces to checking the feasibility of the pair of nodes that had their connecting edge removed. No feasibility check is needed after add since it does not remove any edges, while a delete transformation requires the full k-node connectivity test that is performed on starting configurations.

2.3 Annealing

The annealing schedule decides whether to accept a feasible topologies generated after each transformation. Lower cost topologies are always accepted, while higher cost ones are accepted according to the Boltzmann probability distribution $P(\Delta E) = exp(-\Delta E/kT)$, where $\Delta E$ is the difference in cost between the new and old topologies. Figure 4 summarizes our annealing algorithm and Section 3 discusses aspects of tuning the annealing schedule.
boolean check_feasibility(matrix topology, int N, int connectivity)
{
    for all nodes i=1..N or no_connected == connectivity
    {
        for every other node j {
            if (disjoint_paths(starting_topology, i, j) < connectivity)
                then break;
            else no_connected = no_connected + 1;
        }
        return (no_connected > connectivity)
    }
}

int disjoint_paths(G, s, t)
{
    initialize residual network R to graph G;
    while (shortest_path(new_path, s, t, R)) {
        extract new_path from R;
        paths = paths + 1;
    }
    return(paths);
}

2.3.1 Objective Functions

Recall that our goal is to generate topologies over which replicas of an Internet information service can efficiently and reliably propagate updates among themselves. To meet this goal, besides being 2- or 3-connected, the graphs we generate have low edge-cost and low diameter. This implies that the objective function that controls the annealing schedule combines both edge-cost and diameter. After a transformation, we compute the cost difference $\triangle E$ as

$$\triangle E = p \frac{\text{edge\_cost}_{\text{new}} - \text{edge\_cost}_{\text{old}}}{\text{edge\_cost}_{\text{old}}} + (1-p) \frac{\text{diameter}_{\text{new}} - \text{diameter}_{\text{old}}}{\text{diameter}_{\text{old}}}.$$

Note that this is a linear combination of normalized edge and diameter costs. Choosing $p$ to emphasize edge costs, results in higher diameter graphs; choosing $p$ to emphasize diameter, results in graphs with more and possibly more expensive edges. We compute diameter using Dijkstra’s all-pairs shortest-path algorithm [3] at cost $O(NE\log N)$.

3 Results

This section summarizes experiments we conducted to evaluate the performance of the algorithm presented in the last section. Recall that this algorithm requires a connectivity requirement $k$ as well as an edge-cost matrix of the topology.

We evaluated our algorithm using random graphs generated by NTG [5], a network topology generator. Links were of three different bandwidths; nodes were placed in an x-y plane, with higher density towards the periphery of the plane; and all nodes had the same degree. Link cost was computed as physical distance divided by link bandwidth.

3.1 Annealing Schedule

**Initial Temperature $T_0$:** For each graph $G$, we set the initial $T_0$ by estimating $\triangle E = \triangle E(p)$. We then set the initial temperature $T_0$ so that $P(\triangle E) = \exp(-\triangle nE/kT_0) = 1/2$. Note that different objective functions result in different values of $T_0$.

To study how $T_0$ affects the annealing process, we ran the algorithm for a wide range of values: $T_0/10, T_0,$
and $10T_0$. For a 50-node graph where the feasible topology is 2-connected, the graphs in the first, second and third column of Figure 5 plot total edge-cost, diameter in edge-cost and diameter in hop count, respectively. Each row in Figure 5 corresponds to a different value of $T_n$. Note this is not identical to generating Hamiltonian cycles, since the topologies can have higher degree nodes. The horizontal axis refer to iteration through the annealing process. The topology is accepted when the annealing process terminates, at the far right-hand edge of each graph.

In Figure 5 we observe that as $T_0$ increases, the algorithm allows the objective function to increase, since at high temperatures the annealing process accepts positive $\Delta E$ more frequently. Note that since our initial topologies are nearly cycles, it is hard to eliminate edges without making the resulting graph infeasible. Therefore, early in the the annealing process, edges are added, reducing diameter but increasing edge-cost. Later in the annealing process, both edge-cost and diameter shrink. Note also, that at $T_0$ increases, the diameter and edge-cost fluctuate more and that the topology that minimizes diameter in hop-count is not usually the topology that minimizes diameter in edge-cost.

Figure 6 repeats the previous figure, but for 3-connected topologies. Again, as $T_0$ increases, the annealing process explores higher edge-cost topologies and the objective function oscillations increase. Since 3-connected topologies have more edges than 2-connected ones, their graphs have lower diameters and their individual edges are shorter.

Figure 7 repeats the 2-connected, 50-node experiment with an objective function based only on edge-cost. Notice how the annealing process fails to reduce the diameter significantly. Further note that, at low $T_0$, the algorithm initially fails to find feasible solutions. We indicate this by not showing the objective function lines. While not shown, we repeated the 3-connected experiment with an edge-cost only objective function, and saw similar results.

**Temperature Decrease Rate:** As the annealing progresses, it needs to slowly decrease the temperature until it freezes it. Above, we studied the effects of the initial temperature $T_0$ on the annealing process; below we study the multiplicative temperature decrease rate $D$. In the graphs, the temperature hits zero in $10^{5}$, multiplicative steps, $T_{i+1} = D \cdot T_i$. As illustrated in Figure 4, $T$ decreases after a pre-defined number of transformations that resulted in a topology which the annealing process accepted (SUCCESS\_OPS), or after a pre-defined number of transformations, successful or not (TOTAL\_OPS). Currently, SUCCESS\_OPS and TOTAL\_OPS are set to 10 and 100, respectively.

Figure 8 repeats figure Figure 5 except that instead of studying the effects of different $T_0$, it studies how different temperature decrease rates, $D$, affect the annealing process. Since lower decrease rates make the temperature decrease slower, the annealing process accepts higher-cost topologies more frequently for a longer
anneal() 
{
    for i=1 to ITERATIONS {
        old_cost = current_cost;
        select local_transformation;
        new_topology = local_transformation(current_topology);
        feasible = check_feasibility(new_topology, N, connectivity);
        if (!feasible) continue;
        current_cost = cost(new_topology);
        delta_cost = current_cost - old_cost;
        p = exp(-obj_function(delta_cost)/temperature);
        r = random(0,1);
        if (r < p) {
            accept new_topology;
            success = success + 1;
        } else
        undo transformation;
        operation = operation + 1;
        if ((success == SUCCESS_OPS) or (operations == TOTAL_OPS)) {
            temperature = temperature * (1-decrease_rate);
            success = 0;
            operations = 0;
        }
    }
}

Figure 4: The annealing algorithm.

period of time during the annealing process. We notice
that on average 1, lower decrease rates result in topolo-
gies with lower diameter and slightly higher edge-cost.

While not shown, we repeated the decrease rate ex-
periment with an edge-cost only objective function, and
observed the opposite effect, that is, higher decrease
rates produce topologies with lower total edge-cost, and
slightly higher diameter.

Based on these preliminary results, we set $T_0$ and
$D$ for the experiments described in Section 3.2 below.
We use $T_0$’s lowest value, $T_0 = 0.08$, since it causes the
simulated annealing algorithm to converge sooner. We
use both $D = 0.01$ and $D = 0.4$.

3.2 Edge-Cost versus Diameter

In the next set of simulations, we study how the prob-
ability of applying the add, delete, x-change, and split
transformations affects the annealing process. Besides
driving cost only ($p = 1$) and an equal combination of edge-
cost and diameter in edge-cost ($p = 0.5$), we also use an
equal linear combination of edge-cost and diameter in
hop-count ($p = 1$) as objective functions. For these ex-
periments, we used $P_1 = (50\%, 20\%, 15\%, 15\%)$, $P_2 =
(30\%, 40\%, 15\%, 15\%)$, and $P_3 = (20\%, 50\%, 15\%, 15\%)
$ as the different probability combinations of applying the
add, delete, x-change, and split transformations, respec-
tively. We fixed the initial 50-node topology the anneal-
ing process uses, so that we can compare the resulting
topologies’ costs.

Figure 9 and Table 1 show the results of running the
topology computation algorithm using an equal linear
combination of total edge-cost and diameter in edge-
cost in the objective function to generate 50-node, 2-
connected graphs.

From Table 1, we observe that because the initial
topology already has low total edge-cost, the annealing
process cannot significantly improve total edge-cost, but
produces topologies with diameter up to 70% lower. As
expected, we get lower total edge-cost topologies with
higher diameter as the probability of delete increases
and the probability of add decreases. Note that using
the delete transformation 50% of the times decreases
the topology’s total edge-cost by 10% and still lowers
the diameter by approximately 67%. We also observe

\footnote{We conducted 3 runs for each value of $D$.}
Figure 5: The effects of $T_0$ on the annealing process. For $T_0 = .08$ (1st. row), $T_0 = .8$ (2nd. row), and $T_0 = 8$ (3rd. row), we plot total edge-cost (1st. column), diameter in edge-cost (2nd. column), and diameter in hop-count (3rd. column), when generating 50-node, 2-connected graphs using an equal linear combination of edge-cost and diameter as the objective function ($p = 0.5$). The horizontal axis refer to iteration through the annealing process (log scale).

that while the results in Table 1 show that high delete probabilities attenuate the effects of different decrease rates $D$, the graphs in Figure 9 show that $D = 0.4$ makes the annealing algorithm converge sooner.

We repeat the same experiment using only total edge-cost in the objective function and report the results in Figure 10 and Table 2. Since the annealing process only tries to optimize total edge-cost, higher delete probabilities result in higher total edge-cost reductions. In fact, according to Table 2, when the delete probability is 50%, the resulting topology's total edge-cost is approximately 25% lower than the initial topology's total edge-cost. Notice that we also get a 30% reduction in diameter. The highest diameter reduction is approximately 50% and as expected, happened when the add and delete probabilities were 50% and 20%, respectively. However, we should point out that except for the cases where the add probability is 50%, the diameter in hop-count did not go down significantly. This means that our algorithm generated topologies with almost the same number of edges than the initial topology, but the selected edges have lower cost.

From the graphs in Figure 10, we observe that the lower decrease rate, $D = 0.01$, generates more cost oscillation than $D = 0.4$. Again, this is due to the fact that the lower $D$ causes the temperature to decrease slower, and consequently allows higher-cost topologies to be accepted more frequently by the annealing process.

Figure 11 and Table 3 reports the results obtained when running our simulated annealing algorithm controlled by an equal linear combination of total edge-cost and diameter in hop-count ($p = 0.5$). For networks where physical link costs are roughly uniform, it may make more sense to improve diameter in hop-count than diameter in edge-cost.

From Table 3, we observe that all of the resulting topologies have higher total edge-costs, but show substantial reductions in diameter in hop-count. Notice that the resulting topologies present lower diameter in hop-count than the topologies obtained when diameter in edge-cost is used in the objective function (Table 1).

For the same transformation probability combination, the 0.4 temperature decrease rate generates lower total edge-cost topologies. As the curves in Figure 11 show, the higher decrease rate does not allow total edge-costs to go very high, since the probability of accepting higher-cost topologies decreases faster.

### 3.3 Summary

Choosing the objective function that controls the annealing process depends on the type of optimization...
Figure 6: How $T_0$ affects the annealing process. For $T_0 = .08$ (1st. row), $T_0 = .8$ (2nd. row), and $T_0 = 8$ (3rd. row), we plot total edge-cost (1st. column), diameter in edge-cost (2nd. column), and diameter in hop-count (3rd. column), when generating 50-node, 3-connected graphs using an equal linear combination of edge-cost and diameter as the objective function ($p = 0.5$). The horizontal axis refer to iteration through the annealing process (log scale).

problem being solved. Replicas of an Internet information service need low-cost, low-diameter 2- or 3-connected graphs for propagating their updates. To generate such graphs, the objective function we use consists of an equally weighed linear combination of total edge-cost and diameter. Because the initial topologies our algorithm generates already have low total edge-cost, the resulting topologies show total edge-cost reductions of up to 10%. However, we achieve diameter reductions of up to 70%.

Using total edge-cost alone in the objective function, generates graphs whose total edge-cost and diameter are, respectively, 25% and 30% lower than the feasible but unoptimized starting topology.

If we want to optimize communication cost and propagation delays in terms of number of network hops, we could use an objective function that combines total edge-cost and diameter in hop-count. The corresponding simulation results showed reductions of up to 70% in diameter in hop-count at the expense of total edge-cost increase of approximately 15%. This is because reductions in diameter in hop-count can only be achieved by adding more edges, while swapping more expensive edges with cheaper ones may result in reductions in diameter in edge-cost.

With respect to running time, it takes approximately 7 minutes for the algorithm to execute 10,000 iterations when generating 50-node, 2-connected topologies on a Sun SparcStation 20.

4 Practical Issues

The results in Section 3 demonstrate that simulated annealing generates small enough edge-cost reductions that in practice, simpler, faster, less optimal algorithms can be used. Below, we describe our experience with some of these more practical algorithms.

4.1 Adding Selected Edges

This algorithm starts by generating a random feasible initial topology using the method described in Section 2.1. Recall that these initial topologies tend to have low total edge-cost since they do not include many extra edges. Therefore, in the next step the algorithm adds a number of extra edges so that it can reduce the graph’s diameter.

We used different edge selection heuristics. In the first experiment, we add edges connecting node pairs whose distance in edge-cost is maximum. In other words,
Figure 7: The effects of $T_0$ on the annealing process. For $T_0 = 200$ (1st. row), $T_0 = 2000$ (2nd. row), and $T_0 = 20000$ (3rd. row), we plot total edge-cost (1st. column), diameter in edge-cost (2nd. column), and diameter in hop-count (3rd. column), when generating 50-node, 2-connected graphs using only edge-cost as objective function, ($p = 1$). The horizontal axis refer to iteration through the annealing process (log scale).

<table>
<thead>
<tr>
<th>Extra Edges</th>
<th>Edge Cost</th>
<th>Diameter in Edge-Cost</th>
<th>Diameter in Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9376</td>
<td>4687</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>11294</td>
<td>3216</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>1417</td>
<td>3099</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>12042</td>
<td>3038</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>12882</td>
<td>2761</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>13632</td>
<td>2675</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>14864</td>
<td>2530</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>15159</td>
<td>2358</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>15706</td>
<td>2328</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>16094</td>
<td>1985</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4: Logical topology costs after adding 10 extra edges to the initial topology. Links are added according to diameter in edge-cost.

As expected, total edge-cost goes up as we add new links, and diameter goes down. Notice that after adding the second link, both diameter in edge-cost and hop-count decrease more than 30% with an increase of less than 15% in total edge-cost. We also observe that as diameter in edge-cost decreases, so does diameter in hop-count. However, every time a new link is added diameter in edge-cost decreases, but diameter in hop-count may stay the same.

In the next experiment, we based our selection criteria in diameter in hop-count; when adding an edge, we inspect the current adjacency matrix, and add a link connecting the first pair of nodes, whose distance is equal to diameter in hop-count. Each entry in Table 5 below shows total edge-cost, diameter in edge-cost, and diameter in hop-count every time a new link is added.

Table 5: Logical topology costs after adding 10 extra edges to the initial topology. Links are added according to diameter in hop-count.

Again, as we add new links both diameter in edge-cost and hop-count go down. Notice that after adding the second link, diameter in hop-count decreases almost 50%, while total edge-cost increases less than 20%.

Finally, in the last experiment, we tried to minimize the increase in total edge-cost each time a new edge is added. So, we use the same edge selection criteria as before, except that we choose the cheapest edge connecting nodes whose distance in hop-count is equal to the diameter in hop-count.

When we compare these results with the values in Table 5, we notice that for the same diameter decrease,
Figure 8: The effects of the temperature decrease rate $D$ on the annealing process. For $D = .4$ (1st. row), $D = .1$ (2nd. row), and $D = .01$ (3rd. row), we plot total edge-cost (1st. column), diameter in edge-cost (2nd. column), and diameter in hop-count (3rd. column), when generating 50-node, 2-connected graphs using an equal linear combination of edge-cost and diameter as objective function ($p = 0.5$). The horizontal axis refer to iteration through the annealing process (log scale).

<table>
<thead>
<tr>
<th>Extra Edges</th>
<th>$\Sigma$ Edge Cost</th>
<th>Diameter in Edge-Cost</th>
<th>Diameter in Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9376</td>
<td>4687</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>10290</td>
<td>4635</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>11184</td>
<td>3774</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>12141</td>
<td>3774</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>12786</td>
<td>3774</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>13676</td>
<td>3328</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>14344</td>
<td>3068</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>14734</td>
<td>2936</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>15352</td>
<td>2936</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>15870</td>
<td>2936</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>16543</td>
<td>2936</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 5: Logical topology costs after adding 10 extra edges to the initial topology. Links are added according to diameter in hop-count.

<table>
<thead>
<tr>
<th>Extra Edges</th>
<th>$\Sigma$ Edge Cost</th>
<th>Diameter in Edge-Cost</th>
<th>Diameter in Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9376</td>
<td>4687</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>9751</td>
<td>4626</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>10508</td>
<td>3254</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>11103</td>
<td>3132</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>11895</td>
<td>2908</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>12527</td>
<td>2642</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>13222</td>
<td>2555</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>14046</td>
<td>2553</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>14921</td>
<td>2553</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>15559</td>
<td>2553</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>15874</td>
<td>2390</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 6: Logical topology costs after adding 10 extra edges to the initial topology. Links are added according to diameter in hop-count. The cheapest link is selected every time.
the total edge-cost increase is lower. For example, after we add the second link we get the same reduction in diameter in hop-count, but only a 10% increase in total edge-cost. We also observe that because we select the cheapest link every time, the reduction in diameter in edge-cost is higher than in the previous experiment.

Instead of using the regular add transformation in the simulated annealing approach, we can add selected edges according to one of the criteria presented above. The disadvantage of using the selected add operation is that it is more expensive than the regular add.

4.2 Deleting Selected Edges

The same way as we use heuristics to choose edges to add, we can select good edges to delete. When choosing an edge to delete, the goal is to lower the topology’s total edge-cost, yet keep the diameter constant. Recall that diameter is the maximum shortest path between any pair of nodes. If we want to keep diameter in edge-cost constant, then we should find a directly connected pair of nodes whose connecting edge-cost is equal or higher than the diameter. Another possibility is to find all pair of nodes that satisfy the above requirements, and choose the most expensive link to delete.

Similarly to selected add, we can use the selected delete operation instead of the normal delete operation in the simulated annealing approach. Like selected add, selected delete is also more expensive than the regular delete.

4.3 Minimum-Spanning Tree with Additional Edges

Currently, our Internet replication system uses a minimum-spanning tree (MST) algorithm to generate the graphs over which replicas propagate their updates. This algorithm computes a minimum-cost tree connecting all nodes, and for each node that does not have the required connectivity degree, it adds the cheapest, unused edge. Adding the extra edges not only reduces the graph’s diameter but also improves the corresponding network’s resilience to link and site failures.

Table 7 shows total edge-cost, diameter in edge-cost, and diameter in hop-count for the 50-node graphs this
5 Conclusion

This paper presented and evaluated a simulated annealing algorithm to construct low edge-cost, low diameter, \( k \)-connected graphs. Graphs with these properties are useful for keeping consistency among replicas of an Internet information service. We use 2- or 3-connectivity for resiliency, low edge-cost for efficient use of the physical network, and low diameter for limiting update propagation delays, and consequently transient replica in-

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Extra Edges & \( \sum \text{Edge Cost} \) & Diameter in Edge-Cost & Diameter in Hops \\
\hline
0 & 5212 & 2702 & 23 \\
1 & 5415 & 2702 & 23 \\
2 & 5584 & 2702 & 23 \\
3 & 5713 & 2702 & 23 \\
4 & 6095 & 2643 & 22 \\
5 & 6224 & 2643 & 21 \\
6 & 6454 & 2643 & 21 \\
7 & 6502 & 2643 & 21 \\
8 & 6775 & 2643 & 21 \\
9 & 6972 & 2643 & 21 \\
10 & 7424 & 1971 & 14 \\
11 & 7555 & 1971 & 14 \\
12 & 7981 & 1838 & 14 \\
13 & 8232 & 1831 & 14 \\
14 & 8354 & 1838 & 13 \\
\hline
\end{tabular}
\caption{Total edge-cost, diameter in edge-cost and in hop-count for the graphs resulting from our MST algorithm algorithm. Each entry shows the costs after adding a new edge to the previous graph.}
\end{table}
consistencies.

Our simulated annealing algorithm simultaneously reduces edge-cost and diameter by 10 and 70%, respectively, over unoptimized, feasible solutions. When minimizing edge-cost alone, our algorithm finds solutions that show 25% edge-cost reduction as well as 30% diameter reduction.

Since only moderate reductions in total edge-cost result from such sophisticated algorithm, in practice, simpler, faster, less optimal algorithms can be used. We presented some of these more practical approaches, including the minimum-spanning tree algorithm with additional edges that our Internet replication service currently uses.

References


<table>
<thead>
<tr>
<th>Probabilities (%) Add, Del, Xch, Spt</th>
<th>Decrease Rate (D)</th>
<th>Total Edge Cost Initial</th>
<th>Total Edge Cost Final</th>
<th>Diameter in Edge-Cost Initial</th>
<th>Diameter in Edge-Cost Final</th>
<th>Diameter in Hop-Count Initial</th>
<th>Diameter in Hop-Count Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>50, 20, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>13158</td>
<td>4755</td>
<td>1336</td>
<td>25</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>9495</td>
<td>4755</td>
<td>1664</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td>30, 40, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>9322</td>
<td>4755</td>
<td>1578</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>9398</td>
<td>4755</td>
<td>1572</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>20, 50, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>8872</td>
<td>4755</td>
<td>1594</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>8660</td>
<td>4755</td>
<td>1580</td>
<td>25</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 1: Logical topology costs before and after annealing. The initial temperature is set at 0.08, and as cost function, we use total edge-cost and diameter in edge-cost.

<table>
<thead>
<tr>
<th>Probabilities (%) Add, Del, Xch, Spt</th>
<th>Decrease Rate (D)</th>
<th>Total Edge Cost Initial</th>
<th>Total Edge Cost Final</th>
<th>Diameter in Edge-Cost Initial</th>
<th>Diameter in Edge-Cost Final</th>
<th>Diameter in Hop-Count Initial</th>
<th>Diameter in Hop-Count Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>50, 20, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>7980</td>
<td>4755</td>
<td>2540</td>
<td>25</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>7736</td>
<td>4755</td>
<td>3144</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>30, 40, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>7528</td>
<td>4755</td>
<td>3683</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>7527</td>
<td>4755</td>
<td>3323</td>
<td>25</td>
<td>23</td>
</tr>
<tr>
<td>20, 50, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>7464</td>
<td>4755</td>
<td>2944</td>
<td>25</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>7418</td>
<td>4755</td>
<td>3289</td>
<td>25</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 2: Logical topology costs before and after annealing. The initial temperature is set at 200, and we use total edge-cost as objective function.

<table>
<thead>
<tr>
<th>Probabilities (%) Add, Del, Xch, Spt</th>
<th>Decrease Rate (D)</th>
<th>Total Edge Cost Initial</th>
<th>Total Edge Cost Final</th>
<th>Diameter in Edge-Cost Initial</th>
<th>Diameter in Edge-Cost Final</th>
<th>Diameter in Hop-Count Initial</th>
<th>Diameter in Hop-Count Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>50, 20, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>11364</td>
<td>4755</td>
<td>1364</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>13378</td>
<td>4755</td>
<td>1953</td>
<td>25</td>
<td>7</td>
</tr>
<tr>
<td>30, 40, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>12303</td>
<td>4755</td>
<td>2156</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>11783</td>
<td>4755</td>
<td>2307</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>20, 50, 15, 15</td>
<td>0.01</td>
<td>9517</td>
<td>11254</td>
<td>4755</td>
<td>2088</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>9517</td>
<td>11155</td>
<td>4755</td>
<td>1973</td>
<td>25</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 3: Logical topology costs before and after annealing. The initial temperature is set at 0.08, and we use total edge-cost and diameter in hop-count as objective function when generating 50-node, 2-connected graphs.