The Symmetric Shortest-Path Table Routing Conjecture

Thomas L. Rodeheffer
Microsoft Research, Silicon Valley

May 24, 2013

Abstract

A symmetric shortest-path table routing is a set of paths between all pairs of nodes in a graph such that the set is closed under path reversal, each path is a shortest path in the graph, and all paths with the same destination form a tree with a sink at the destination. Such a routing can be found for any finite, connected, undirected, positive-weighted graph by "salting" the edge weights so that all the shortest salted paths are unique while each one remains a shortest path in the original graph. A conjecture arises about whether every possible symmetric shortest-path table routing can be selected in this way by some edge weight salting. It turns out that this conjecture is false. This paper describes the search for counterexamples.

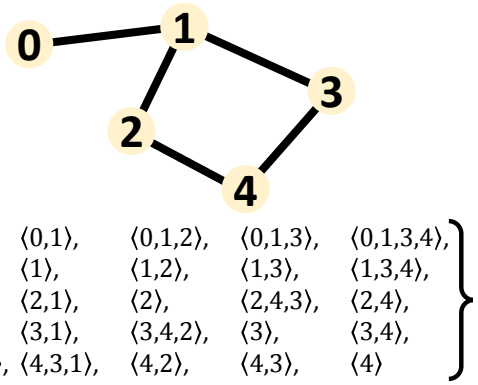


Figure 1: Example graph and routing. The routing contains one path from a to b for each pair of nodes a, b in the graph.

1 Introduction

We consider a finite, connected, undirected, positive-weighted graph G. There is at most one edge between each pair of nodes and no self-loops. In general each edge has an arbitrary positive (non-zero) weight although often we consider just unit weights. We let n be the number of nodes and label the nodes $0, 1, \ldots, n-1$.

Since G is connected, there is a path from a to b for each pair of nodes a, b. A *routing* is a set containing one such path for every pair of nodes. Figure 1 shows an example graph with a routing.

A *shortest-path routing* is a routing in which each path is a shortest path in G. The length of a path is defined as the sum of the weights of its edges.

A *symmetric routing* is a routing which is closed under path reversal. In other words, for each path p in the routing, the reverse of path p is also in the routing.

Each path in a routing for G can be considered as a set of directed edges on the nodes of G. If the set of all paths with a given destination d forms a directed tree with sink d we say that the routing forms a *destination tree* for d.

A *table routing* is a routing which forms a destination tree for every node in G.

2 Network routing rationale

In a network connected by point-to-point links, one common way of arranging communication of messages between arbitrary source and destination nodes is the forwarding table method, which (1) assigns a unique number to each node, (2) attaches the destination node number to each message, and (3) provides each node with a local

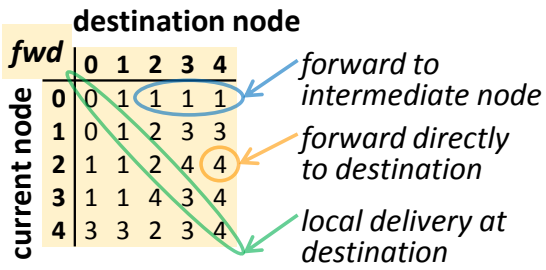


Figure 2: Forwarding table for Figure 1.

forwarding table that maps each destination node number to an appropriate next-hop forwarding action, either forwarding the message to a neighboring node or delivering it locally when it has reached its destination.

Typically, links are bidirectional and links that connect a node back to itself are useless. In this paper we take the simplifying assumption that there are no duplicate links between any pair of nodes and no self-loops from a node back to itself.

This networking scenario can be modeled as a finite, connected, undirected graph with a table routing. Because the forwarding action at each node depends only on the destination of a message, without regard to the origination or past history of the message, the network forwarding of messages for any destination d forms a destination tree. Hence the routing is a table routing.

A table routing can be represented as a global *forwarding table* that specifies the forwarding action $fwd(c, d)$ for each pair of current node c and destination node d. The forwarding action can be represented as a node number: either the neighboring node number b in the case of an action that forwards the message to neighboring node b, or the current node number c in the case of an action that delivers the message locally. Since the network contains no self-loops and no duplicate links, this representation completely specifies the forwarding action.

The example routing illustrated in Figure 1 happens to be a table routing. Figure 2 illustrates its forwarding table.

Note that not all possible forwarding tables correspond to a table routing. Forwarding tables might specify forwarding actions that cannot be performed by any link in the network, that deliver a message to the wrong destination, or that forward a message endlessly without ever delivering it. Such forwarding tables are not *valid*. In this paper we only consider valid forwarding tables, those which correspond to table routings.

A table routing might also be a shortest-path routing. A shortest-path routing can be important in a networking context when shorter paths are more desirable due to lower delay or lower cost. The example shown in Figure 1 happens to be a shortest-path table routing.

A table routing might also be a symmetric routing. A symmetric routing can be important in a networking context in certain control protocols when forward-flowing reservations leave state in intermediate nodes that needs to be compensated by reverse-flowing acknowledgements. The example shown in Figure 1 happens to be a symmetric table routing.

The question arises, is there always a symmetric, shortest-path, table routing for a finite, connected, undirected, positive-weighted graph? The answer is yes, as we next show by construction.

3 The edge weight salting method

Suppose we have a finite, connected, undirected, positive-weighted graph G. The following algorithm shows how to find a symmetric shortest-path table routing for G.

The only difficulty arises in the case of ties in shortest paths. If every shortest path is unique, then for each destination d, the set of all shortest paths ending at d forms a destination tree. Hence we can just take the set of all shortest paths and we will have a table routing for G. Since all of the paths in the routing are shortest-paths, the routing is a shortest-path routing. Since the graph is undirected and the addition of edge weights is commutative, the reverse of a shortest path from a to b is a shortest path from b to a. Hence the routing is a symmetric routing.

Given a graph in which there may be ties in shortest path lengths, we add "salt" to the edge weights so that all shortest salted path lengths are unique, while preserving the condition that any shortest salted path is a shortest path in the original graph. We then take the unique shortest salted paths to obtain a symmetric shortest-path table routing for the salted graph, which is also such a routing for the original graph.

It remains to describe how to salt the edge weights properly. We arbitrarily number the edges $1, 2, \ldots$ and

define w_i as the weight of edge i. Then we define w'_i, the salted weight of edge i, by

$$w'_i = w_i + \left(\frac{\epsilon}{2}\right)^i$$

where $\epsilon > 0$ is a small number chosen as follows.

Consider any pair of nodes a, b. Let $dist(a, b)$ be the length of the shortest path from a to b in the original graph. Consider all finite sums of edge weights, in which each edge weight is permitted to participate any non-negative number of times. Because there are only a finite number of edges and all the weights are positive, there can only be a finite number of sums less than $dist(a, b) + 1$. Hence we can choose an $\epsilon_{a,b} > 0$ such that there are no sums σ such that

$$0 < |dist(a, b) - \sigma| < \epsilon_{a,b}$$

We choose ϵ as the minimum $\epsilon_{a,b}$ over all a, b. This minimum exists because the number of nodes is finite.

Hence there is no sum of edge weights, in particular there is no path length, that falls within ϵ of a shortest path length without being identical to the shortest path length.

We generalize w and w' to compute the lengths of paths by summing up the edge weights or salted edge weights, respectively.

To prove that any shortest path in the salted graph is some shortest path in the original graph, consider any pair of nodes a, b. Let p, q be paths from a to b where p is a shortest path in the original graph and q is not. Observe that

$$w_p = dist(a, b)$$
$$w'_q \geq w_q \geq dist(a, b) + \epsilon$$

Since no shortest path can repeat an edge and each edge number is a non-zero natural, we must have

$$w'_p = \sum_{i \in p} w'_i = \sum_{i \in p} \left(w_i + \left(\frac{\epsilon}{2}\right)^i\right) < dist(a, b) + \epsilon$$

Hence we have $w'_p < w'_q$. It follows that any shortest path in the salted graph is some shortest path in the original graph.

To prove that the salted graph has no shortest path ties, consider any pair of nodes a, b and suppose we have two shortest paths p and q from a to b in the salted graph, $w'_p = w'_q$. Since each shortest path in the salted graph is also a shortest path in the original graph, we have $w_p = w_q = dist(a, b)$. It follows that

$$\sum_{i \in p} \left(\frac{\epsilon}{2}\right)^i = \sum_{j \in q} \left(\frac{\epsilon}{2}\right)^j$$

Since no shortest path can repeat an edge, we can determine the set of edges from the powers of $\epsilon/2$ and conclude that both p and q contain the same set of edges. If this set is empty, the paths are the same. Assuming the set is not empty, then since both paths must start at node a and since no shortest path can repeat a node, there can be only one edge in the set that starts at a and consequently both paths must start with the same edge. By induction, the paths are the same. Hence the salted graph has only one shortest path from a to b.

Since a sufficiently small ϵ exists, an implementation need not compute it, but can maintain it symbolically using $w'_i = \langle w_i, 1/2^i \rangle$. By adding such pairs componentwise and comparing them lexicographically, the implementation will get the correct answer for any comparison of the length of a shortest salted path against the length of a non-shortest salted path. Hence an implementation can use lengths expressed as these pairs to compute the set of shortest salted paths.

4 The conjecture

Given the edge weight salting method for constructing a symmetric, shortest-path, table routing, it seems natural to ask if the method is capable of constructing *any* such desired routing by using an appropriate salt. This leads to the following conjecture:

Conjecture 1 *Given a finite, connected, undirected, positive-weighted graph G, and a symmetric shortest-path table routing fwd for G, there exists some assignment of edge weight salts such that the edge weight salting method computes fwd as its result.*

It turns out that the conjecture is *false*, as we show in Section 6. First we show that it suffices to search for counterexamples among graphs and routings using unit edge weights.

5 Reduction to unit edge weights

5.1 Real to rational

Suppose there is a counterexample using positive real edge weights. From this counterexample we construct a counterexample using positive rational edge weights, as follows.

We partition the edge weights into mutually incommensurate subsets. If there is only one subset, then all of the edge weights are commensurate and by choosing one weight v, we can produce a rational counterexample by dividing all the weights by v. Scaling all the edge weights scales all of the path lengths but does not change any decision regarding shortest paths.

Otherwise, we choose two subsets W_1, W_2 and two representative weights $v_1 \in W_1$, $v_2 \in W_2$. The idea is to scale the weights in W_1 by a tiny factor so that the resulting weights are commensurate with v_2, without altering any decision regarding shortest paths. This produces a new counterexample with a reduced number of subsets in the partition. By induction, we can produce a rational counterexample.

Since each weight in W_1 is incommensurate with any weight not in W_1, any two paths of equal length must include the same number of edges with weights from W_1. So, if all of the weights in W_1 are scaled by the same factor, these two paths will still have equal length. In particular, any shortest path ties before scaling will still be tied in length after scaling, although they might not remain as shortest paths. It remains to determine a factor small enough to ensure that they remain as shortest paths.

Consider any pair of nodes a, b. Recall that $dist(a, b)$ is the length of the shortest path from a to b. Consider all finite sums of edge weights, in which each edge weight is permitted to participate any non-negative number of times. Because there are only a finite number of edges and all the weights are positive, there can only be a finite number of sums less than $dist(a, b) + 1$. Hence we can choose an $\epsilon_{a,b} > 0$ such that there are no sums σ such that

$$0 < |dist(a, b) - \sigma| < \epsilon_{a,b}$$

Furthermore, we can define $m_{a,b}$ as the maximum number of terms in any edge weight sum less than $dist(a, b)+1$. Based on these values, we define $\delta_{a,b} > 1$ as

$$\delta_{a,b} = 1 + \frac{\epsilon_{a,b}}{m_{a,b}}$$

Observe that if each edge weight in W_1 is scaled by any factor in the interval $(1 \,..\, \delta_{a,b})$, then each edge weight sum will increase by some amount, and each edge weight sum less than $dist(a, b)+1$ will increase by less than $\epsilon_{a,b}$. Hence any shortest path from a to b will remain a shortest path.

We choose $\delta > 1$ as the minimum $\delta_{a,b}$ over all a, b. This minimum exists because the number of nodes is finite. Observe that if each edge weight in W_1 is scaled by any factor in the interval $(1 \,..\, \delta)$, each shortest path will remain a shortest path.

We chose q as any rational number

$$q \in \left(\frac{v_1}{v_2} \,..\, \delta\frac{v_1}{v_2}\right)$$

Since $\delta > 1$, this interval has non-zero width and hence contains a rational number. Observe that

$$q\frac{v_2}{v_1} \in (1 \,..\, \delta)$$

Hence, scaling each edge weight in W_1 by qv_2/v_1 does not change the set of shortest paths. Since q is a rational number, this makes all of the edge weights in W_1 commensurate with v_2, as desired.

5.2 Rational to natural

Suppose there is a counterexample using positive rational edge weights. From this counterexample we construct a counterexample using positive natural edge weights, as follows.

Since each edge weight is a positive rational number, it can be expressed as a natural numerator and a natural denominator. Let m be the least common multiple of all the denominators.

We produce a natural counterexample by multiplying all the edge weights by m. Scaling all the edge weights scales all of the path lengths but does not change any decision regarding shortest paths.

5.3 Natural to unit

Suppose there is a counterexample using positive natural edge weights. From this counterexample we construct a counterexample using unit edge weights, as follows.

For edge number i, let w_i be the weight of edge i. We create a new graph by "exploding" this edge into w_i consecutive unit weight edges, inserting $w_i - 1$ new intermediate nodes.

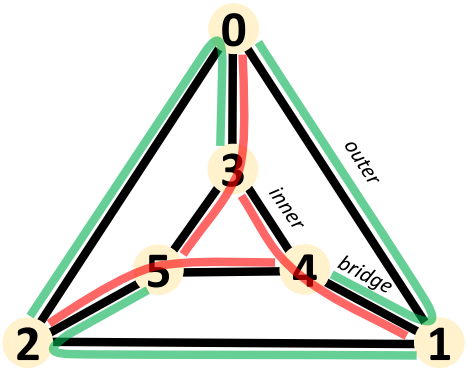
Repeating this process for all edges produces a new graph containing only unit weight edges. Any shortest path in the original graph is a shortest path in the new graph. If there is a salting of the unit edges in the new graph that produces the required table routing, then we can produce a corresponding salting of the original graph by simply adding up the salts on each of the exploded edges. Hence if the original graph is a counterexample, so must also be the new graph.

6 First counterexample

Lyle Ramshaw found the first counterexample to the conjecture. His counterexample uses six nodes, nine edges, and unit edge weights. Figure 3 shows an illustration of the counterexample.

The counterexample is constructed using six nodes, the first three nodes forming an outer triangle with three outer edges, the second three nodes forming an inner triangle with three inner edges, and three bridge edges connecting the inner and outer triangles. Since the edges are unit weight, path length is equivalent to hop count.

Shortest paths in the graph are either one-hop or two-hop. Each of the one-hop shortest paths corresponds to a single edge in the graph and is unique. The two-hop shortest paths connect nodes that are at distance two. In each of these cases, one node is found on the outer triangle and the other on the inner triangle and there are two shortest paths available. When the destination is located clockwise of the source, the routing first takes an edge that remains on the source triangle then subsequently takes a bridge edge to arrive at the destination. When the destination is located counterclockwise of the source, the routing first takes a bridge edge to change triangles then subsequently takes an edge that remains on the destination triangle to arrive at the destination.



Edges are shown in black and are one-hop routes.
Two-hop routes are shown in color.

	0	1	2	3	4	5
0	-	X	X	X	-	-
1	X	-	X	-	X	-
2	X	X	-	-	-	X
3	X	-	-	-	X	X
4	-	X	-	X	-	X
5	-	-	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	1	3
1	0	1	2	4	4	2
2	0	1	2	0	5	5
3	0	4	0	3	4	5
4	1	1	5	3	4	5
5	3	2	2	3	4	5

Forwarding table

Figure 3: Lyle Ramshaw's counterexample.

Clearly, the routing is a shortest-path routing. Since each two-hop route ends in a one-hop route, the routing forms a destination tree for each destination and consequently is a table routing. By inspection, we see that the reverse of each route is also a route. Hence the routing is a symmetric routing.

It remains to show that no possible edge weight salting can produce this routing. We proceed by assuming that a suitable salting exists and derive a contradiction. For any pair of adjacent nodes a, b, let $salt_{a,b}$ be a suitable salting of edge $\langle a, b\rangle$. Since edges are undirected, we have $salt_{a,b} = salt_{b,a}$. For uniformity, we list the edge using the lower-numbered node first.

Observe that two shortest paths are available for each pair of nodes at distance two. For example, there are two shortest paths that connect node 0 to node 4: $\langle 0, 1, 4\rangle$ and $\langle 0, 3, 4\rangle$. Because node 4 is clockwise from node 0, the routing chooses the path that takes the outer edge $\langle 0, 1\rangle$ followed by the bridge edge $\langle 1, 4\rangle$ as opposed to the other path. In order for path $\langle 0, 1, 4\rangle$ to have a shorter salted

length that path $\langle 0, 3, 4\rangle$, we must have

$$salt_{0,1} + salt_{1,4} < salt_{0,3} + salt_{3,4}$$

The clockwise routes from the two other outer triangle nodes have similar situations, resulting in the inequalities

$$salt_{1,2} + salt_{2,5} < salt_{1,4} + salt_{4,5}$$

$$salt_{0,2} + salt_{0,3} < salt_{2,5} + salt_{3,5}$$

In summary, for each of the three two-hop clockwise routes starting on the outer triangle (illustrated in green in Figure 3), we have an inequality that says that an outer edge salt plus a bridge edge salt must be less than a bridge edge salt plus an inner edge salt. Adding up these three inequalities, the bridge salts cancel out, resulting in an inequality that says that the sum of the outer edge salts must be less than the sum of the inner edge salts:

$$salt_{0,1} + salt_{1,2} + salt_{0,2} < salt_{3,4} + salt_{4,5} + salt_{3,5}$$

However, a complementary situation holds with respect to the three two-hop clockwise routines starting on the inner triangle (illustrated in red in Figure 3). For each of these three routes, we have an inequality that says that an inner edge salt plus a bridge edge salt must be less than a bridge edge salt plus an outer edge salt. Adding up these three inequalities, the bridge salts cancel out, resulting in an inequality that says that the sum of the inner edge salts must be less than the sum out the outer edge salts:

$$salt_{3,4} + salt_{4,5} + salt_{3,5} < salt_{0,1} + salt_{1,2} + salt_{0,2}$$

But it is impossible that the sum of the outer edge salts can be both less than and greater than the sum of the inner edge salts. Hence there is no possible edge weight salting that can produce this routing.

7 Exhaustive search

Given that a unit edge weight counterexample was found amoung graphs with six nodes, we wondered if there were any smaller counterexample.

We had not initially tried to use exhaustive search to find counterexamples because (1) the search space grows enormously with increasing model parameters, generally making it infeasible to search much beyond trivial model sizes and (2) we thought the conjecture was probably true anyway. Well, the conjecture was now proven false, and six nodes was not too inconceivably large of a graph size. So we decided to try an exhaustive search program.

7.1 Enumeration of routings

The program operates as follows, in order to search all graphs of n nodes. First, it enumerates all $2^{n(n-1)/2}$ possible edge combinations and discards those that result in graphs that are not connected. The connected graphs are filtered to remove isomorphisms.

Removing isomorphisms requires keeping a list of all the distinct instances so far encountered, against which all possible node permutations have to be checked. Although comparing the distribution of node degrees can often help skip a lot of this checking, the process is still fairly expensive. However, it is a huge net win, because there are many isomorphisms and removing the duplicates means that they do not have to be processed in any later stage.

For each distinct connected graph, the program goes through each destination node b and computes all shortest paths ending at b. The entire set of possible sortest paths does not need to be computed explicitly, because it suffices just to determine at each node and for each destination what next hop nodes lie on shortest paths to that destination. This is easily done by a breadth-first search graph walk starting at the destination. The result determines for each current node a and destination node b, the set of possible forwarding table values for $fwd(a, b)$ that are consistent with forwarding on some shortest path from a to b.

For example, looking at the graph in Figure 3, there are two shortest paths from node 0 to node 4: $\langle 0, 1, 4\rangle$ and $\langle 0, 3, 4\rangle$. These paths differ in the forwarding action at node 0; the options can be summarized as $fwd(0, 4) \in \{1, 3\}$. Table 1 lists all of the shortest-path forwarding options for this graph.

For each pair of nodes a, b we define $opt(a, b)$ as the set of shortest-path forwarding options for $fwd(a, b)$.

Selecting any combination of options results in a forwarding table that specifies a shortest-path table routing on the selected graph. Furthermore, this process enumerates all possible shortest-path table routings. The program checks each routing for closure under path reversal and

current node \ destination node	0	1	2	3	4	5
0	$\{0\}$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,3\}$	$\{2,3\}$
1	$\{0\}$	$\{1\}$	$\{2\}$	$\{0,4\}$	$\{4\}$	$\{2,4\}$
2	$\{0\}$	$\{1\}$	$\{2\}$	$\{0,5\}$	$\{1,5\}$	$\{5\}$
3	$\{0\}$	$\{0,4\}$	$\{0,5\}$	$\{3\}$	$\{4\}$	$\{5\}$
4	$\{1,3\}$	$\{1\}$	$\{1,5\}$	$\{3\}$	$\{4\}$	$\{5\}$
5	$\{2,3\}$	$\{2,4\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$

Table 1: Shortest-path forwarding options for the graph in Figure 3.

discards those that are not symmetric. The symmetric shortest-path table routings are filtered to remove isomorphisms.

Table routings fwd_1, fwd_2 for the same graph are isomorphic iff a permutation π of the nodes induces an automorphism on the graph and for all nodes a, b we have

$$\pi(fwd_1(a,b)) = fwd_2(\pi(a), \pi(b))$$

Removing isomorphisms requires keeping a list of all the distinct instances so far encountered. Although the filtering is fairly expensive, it is a huge net win, because typically each table routing has many automorphisms.

The result is an enumeration of all possible distinct symmetric shortest-path table routings on the selected graph. Each of these table routings are considered in turn as a "desired" table routing. We use $fwd(a,b)$ to refer to the desired forwarding table and we define $path(a,b)$ as the desired path from a to b.

7.2 Edge salt constraints

Next the search program must determine if there exists an edge weight salting that would produce the desired table routing. Basically, this salting must result in choosing the desired $fwd(a,b) \in opt(a,b)$ for each a, b.

Consider the problem in stages by increasing hop count and see what constraints can be deduced. For zero-hop shortest paths, $opt(a,b)$ is always a singleton, because at the destination and $a = b$ and $opt(a,b) = \{b\}$. So there is no constraint on the edge weight salting that results from zero-hop shortest paths.

Now consider what can be deduced for shortest paths of $i+1$ hops given that all constraints for shortest paths up to i hops have been deduced. Let a, b be a node pair with an $i+1$ hop shortest path from a to b. Let $f = fwd(a,b)$, the desired forwarding from a with destination b. We know that $f \in opt(a,b)$.

In order for the edge salts to cause f to be selected out of $opt(a,b)$, it must be the case that some shortest path $\langle a, f, \ldots, b\rangle$ has a lower salted path length than every shortest path $\langle a, c, \ldots, b\rangle$ for every $c \in opt(a,b)$, $c \neq f$. Since all shortest paths from a to b have the same hop count, we can cancel out the unit edge weights from the salted path lengths and concentrate solely on the sum of the edge salts.

Since the shortest paths from f to b and the shortest paths from c to b are all i-hop shortest paths, sufficient contraints on the edge salts have already been deduced to guarantee that the desired shortest paths are selected. So it only remains to deduce whatever additional edge salt constraints are necessary to prefer f over each $c \in opt(a,b)$, $c \neq f$. For each such c, it must be the case that

$$salt_{a,f} + \sum_{k \in path(f,b)} salt_k < salt_{a,c} + \sum_{k \in path(c,b)} salt_k$$

Here we use $salt_k$ to refer to the salt of edge k.

By going through all the shortest-path forwarding options in this way, the search program collects a system of simultaneous inequalities that must be satisfied by the edge salts in order to produce the desired forwarding table. It remains to determine if this system has a solution.

7.3 Solving the constraints

Observe that each constraint in the system consists of an inequality with a sum of an equal number of edge salts on each side. Hence, since there are only a finite number of edge salts, if the system has a solution in real numbers, the solution can be translated into a solution in positive reals.

A very powerful method for solving systems of inequalities over positive reals is the Dantzig simplex algorithm [2, 4]. However, we have to make a couple of adjustments in order to apply it.

First, the simplex algorithm requires inequalities to be expressed as $\leq$ but the constraints are expressed using $<$.

Fortunately, we can reformulate each constraint from

$$\sum_j salt_j < \sum_k salt_k$$

to

$$1 + \sum_j salt_j \leq \sum_k salt_k$$

Obviously, any solution to the reformulated constraints satisfies the original constraints. Fuirthermore, if there exists any solution to the original constraints, then by scaling it sufficiently, we can produce a solution to the reformulated constraints. Hence, from the point of view of searching for counterexamples, the reformulated system is equivalent to the original system.

Second, the simplex algorithm requires starting with a feasible solution, which it then proceeds to transform into a feasible solution that minimizes a linear cost function. But our entire problem is to find any feasible solution. Fortunately, this problem is addressed by the two-phase simplex algorithm, which in phase one introduces an artificial variable into each inequality in order to make it trivial to produce the initial feasible solution. The simplex algorithm is then applied to minimize a cost function that reflects the presence of the artificial variables. If the optimal value of the cost function is zero, then a feasible solution has been obtained in which the artificial variables take no part.

When the simplex algorithm terminates, it has either found an optimum solution or else has determined that no optimum solution exists. If the optimum solution has not reduced the cost to zero, then there is no solution to the constrains that does not involve artificial variables, so it cannot be solved as originally written.

The search program uses an implementation of the phase one simplex algorithm written using rational arithmetic, so if it produces a solution the solution is exact. For purposes of debugging, the program takes this solution, uses the edge salt method to determine the resulting forwarding table, and verifies that the result is indeed the desired forwarding table. (This check uncovered a number of bugs during the development of the search program.)

On the other hand, if the phase one simplex algorithm says that there is no solution, the search program has found a counterexample. In this case, the counterexample is logged.

nodes	distinct connected graphs	distinct symmetric shortest-path table routings	counter-examples	runtime (sec)
1	1	1	0	0
2	1	1	0	0
3	2	2	0	0
4	6	6	0	0
5	21	36	0	0
6	112	680	17	6
7	853	60943	3642	17420

Table 2: Results of the exhaustive search.

7.4 Results of the search

We wrote the exhaustive search program in C# and ran it for graphs up through seven nodes on an Intel® Xeon® CPU W3550 with 12 GB of memory running at 3.07 GHz. Table 2 lists the results. The results clearly show the enormous explosion of the search space as the size of the graphs increase.

No counterexamples were found for any graphs smaller than six nodes. 17 counterexamples were found among graphs with six nodes and 3642 among graphs with seven nodes. Appendix A lists the 17 six-node counterexamples. Lyle Ramshaw's original counterexample appears as number 7.

By reasoning analogous to the reductions in Sections 5.1 and 5.2, if any solution among positive real numbers exists to the system of simultaneous inequalities, then there exists a solution among the naturals. Hence each counterexample is equivalent to a theorem that the system cannot be solved among the naturals.

To check that the search program was correct in determining that these theorems are true, we had the search program translate each six-node counterexample into a TLA+[3] theorem. Appendix B lists these theorems.

Each of these theorems was directly verified by the Z3 SMT solver, one of the back-end provers available to the TLA Proof System [1]. SMT solvers are particularly good at verifying facts involving linear combinations of naturals. The entire verification took less than one minute of runtime.

8 Conclusion

The symmetric shortest-path table routing conjecture admits of fairly simple counterexamples, which could have been found easily enough by exhaustive search. In fact, even our initial, rough search program found counterexamples in less than four minutes; after filtering isomorphisms, the final search program finds them in a few seconds.

However, while the conjecture was still open, we did not put any effort into performing an exhaustive search because we were convinced that it would have been fruitless. The lesson to learn is that one should put equal effort into both sides of a conjecture.

The counterexamples listed in Appendix A clearly seem to fall into a few main categories. It remains future work to see if any interesting classifications can be made.

Acknowledgements

Thanks to Lyle Ramshaw, Andrew Goldberg, Mark Manasse, and Sergey Yekhanin, who at various times provided input on the conjecture. Thanks to Mike Schroeder, Rama Kotla, and George Varghese for asking provocative questions.

References

[1] K. Chaudhuri, D. Doligez, L. Lamport, and S. Merz. The TLA+ proof system: Building a heterogeneous verification platform. In *Proceedings of the 7th International colloquium conference on Theoretical aspects of computing*, ICTAC'10, pages 44–44, Berlin, Heidelberg, 2010. Springer-Verlag.

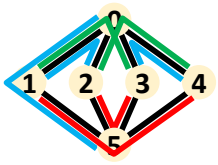
[2] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Aug. 1998.

[3] L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

[4] J. C. Nash. The (Dantzig) simplex method for linear programming. *Computing in Science and Engineering*, 2(1):29–31, Jan. 2000.

A Counterexamples (n=6)

Counterexample 1:



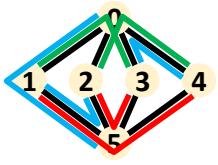
	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	-	-	-	X
2	X	-	-	-	-	X
3	X	-	-	-	-	X
4	X	-	-	-	-	X
5	-	X	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	1
1	0	1	0	0	5	5
2	0	0	2	5	0	5
3	0	0	5	3	0	5
4	0	5	0	0	4	5
5	1	1	2	3	4	5

Forwarding table

Counterexample 2:



	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	-	-	-	X
2	X	-	-	-	-	X
3	X	-	-	-	-	X
4	X	-	-	-	-	X
5	-	X	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	1
1	0	1	5	0	5	5
2	0	5	2	5	0	5
3	0	0	5	3	0	5
4	0	5	0	0	4	5
5	1	1	2	3	4	5

Forwarding table

Counterexample 3:

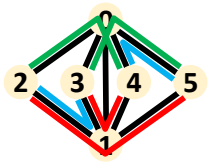
	0	1	2	3	4	5
0	-	X	X	X	X	X
1	X	-	X	X	X	X
2	X	X	-	-	-	-
3	X	X	-	-	-	-
4	X	X	-	-	-	-
5	X	X	-	-	-	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	0	1	2	3	4	5
2	0	1	2	0	0	1
3	0	1	0	3	1	0
4	0	1	0	1	4	0
5	0	1	1	0	0	5

Forwarding table

Counterexample 4:



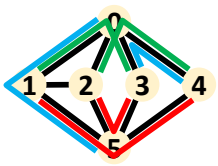
	0	1	2	3	4	5
0	-	X	X	X	X	X
1	X	-	X	X	X	X
2	X	X	-	-	-	-
3	X	X	-	-	-	-
4	X	X	-	-	-	-
5	X	X	-	-	-	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	0	1	2	3	4	5
2	0	1	2	1	0	1
3	0	1	1	3	1	0
4	0	1	0	1	4	0
5	0	1	1	0	0	5

Forwarding table

Counterexample 5:



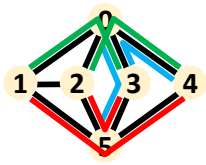
	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	X	-	-	X
2	X	X	-	-	-	X
3	X	-	-	-	-	X
4	X	-	-	-	-	X
5	-	X	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	1
1	0	1	2	0	5	5
2	0	1	2	5	0	5
3	0	0	5	3	0	5
4	0	5	0	0	4	5
5	1	1	2	3	4	5

Forwarding table

Counterexample 6:



	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	X	-	-	X
2	X	X	-	-	-	X
3	X	-	-	-	-	X
4	X	-	-	-	-	X
5	-	X	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	3
1	0	1	2	0	5	5
2	0	1	2	5	0	5
3	0	0	5	3	0	5
4	0	5	0	0	4	5
5	3	1	2	3	4	5

Forwarding table

Counterexample 7:

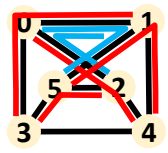
	0	1	2	3	4	5
0	-	X	X	X	-	-
1	X	-	X	-	X	-
2	X	X	-	-	-	X
3	X	-	-	-	X	X
4	-	X	-	X	-	X
5	-	-	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	1	3
1	0	1	2	4	4	2
2	0	1	2	0	5	5
3	0	4	0	3	4	5
4	1	1	5	3	4	5
5	3	2	2	3	4	5

Forwarding table

Counterexample 8:



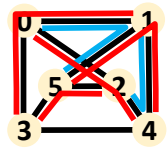
	0	1	2	3	4	5
0	-	X	X	X	-	-
1	X	-	-	-	X	X
2	X	-	-	-	X	X
3	X	-	-	-	X	X
4	-	X	X	X	-	-
5	-	X	X	X	-	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	2	1
1	0	1	0	0	4	5
2	0	0	2	5	4	5
3	0	0	5	3	4	5
4	2	1	2	3	4	1
5	1	1	2	3	1	5

Forwarding table

Counterexample 9:



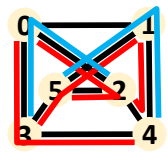
	0	1	2	3	4	5
0	-	X	X	X	-	-
1	X	-	-	-	X	X
2	X	-	-	-	X	X
3	X	-	-	-	X	X
4	-	X	X	X	-	-
5	-	X	X	X	-	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	2	1
1	0	1	4	0	4	5
2	0	4	2	5	4	5
3	0	0	5	3	4	5
4	2	1	2	3	4	1
5	1	1	2	3	1	5

Forwarding table

Counterexample 10:



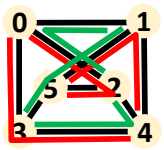
	0	1	2	3	4	5
0	-	X	X	X	-	-
1	X	-	-	-	X	X
2	X	-	-	-	X	X
3	X	-	-	-	X	X
4	-	X	X	X	-	-
5	-	X	X	X	-	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	3	2
1	0	1	4	5	4	5
2	0	4	2	0	4	5
3	0	5	0	3	4	5
4	3	1	2	3	4	1
5	2	1	2	3	1	5

Forwarding table

Counterexample 11:



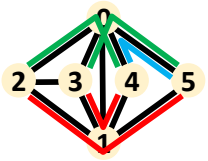
	0	1	2	3	4	5
0	-	X	X	X	-	-
1	X	-	-	-	X	X
2	X	-	-	-	X	X
3	X	-	-	-	X	X
4	-	X	X	X	-	-
5	-	X	X	X	-	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	3	2
1	0	1	0	5	4	5
2	0	0	2	4	4	5
3	0	5	4	3	4	5
4	3	1	2	3	4	1
5	2	1	2	3	1	5

Forwarding table

Counterexample 12:



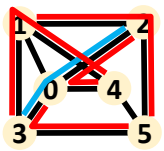
	0	1	2	3	4	5
0	-	X	X	X	X	X
1	X	-	X	X	X	X
2	X	X	-	X	-	-
3	X	X	X	-	-	-
4	X	X	-	-	-	-
5	X	X	-	-	-	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	0	1	2	3	4	5
2	0	1	2	3	0	1
3	0	1	2	3	1	0
4	0	1	0	1	4	0
5	0	1	1	0	0	5

Forwarding table

Counterexample 13:



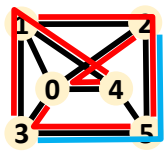
	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	X	X	X	-
2	X	X	-	-	-	X
3	X	X	-	-	-	X
4	X	X	-	-	-	X
5	-	-	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	3
1	0	1	2	3	4	2
2	0	1	2	0	0	5
3	0	1	0	3	1	5
4	0	1	0	1	4	5
5	3	2	2	3	4	5

Forwarding table

Counterexample 14:



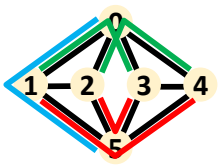
	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	X	X	X	-
2	X	X	-	-	-	X
3	X	X	-	-	-	X
4	X	X	-	-	-	X
5	-	-	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	3
1	0	1	2	3	4	2
2	0	1	2	5	0	5
3	0	1	5	3	1	5
4	0	1	0	1	4	5
5	3	2	2	3	4	5

Forwarding table

Counterexample 15:



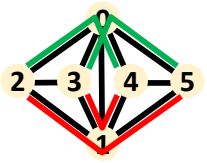
	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	X	-	-	X
2	X	X	-	-	-	X
3	X	-	-	-	X	X
4	X	-	-	X	-	X
5	-	X	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	1
1	0	1	2	0	5	5
2	0	1	2	5	0	5
3	0	0	5	3	4	5
4	0	5	0	3	4	5
5	1	1	2	3	4	5

Forwarding table

Counterexample 16:



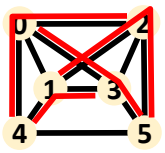
	0	1	2	3	4	5
0	-	X	X	X	X	X
1	X	-	X	X	X	X
2	X	X	-	X	-	-
3	X	X	X	-	-	-
4	X	X	-	-	-	X
5	X	X	-	-	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	0	1	2	3	4	5
2	0	1	2	3	0	1
3	0	1	2	3	1	0
4	0	1	0	1	4	5
5	0	1	1	0	4	5

Forwarding table

Counterexample 17:



	0	1	2	3	4	5
0	-	X	X	X	X	-
1	X	-	X	X	X	-
2	X	X	-	X	-	X
3	X	X	X	-	-	X
4	X	X	-	-	-	X
5	-	-	X	X	X	-

Adjacency matrix

	0	1	2	3	4	5
0	0	1	2	3	4	3
1	0	1	2	3	4	2
2	0	1	2	3	0	5
3	0	1	2	3	1	5
4	0	1	0	1	4	5
5	3	2	2	3	4	5

Forwarding table

B Counterexample theorems (n=6)

MODULE $NoSalt6$

EXTENDS $Naturals$, $TLAPS$

THEOREM $Cx1 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s0t4,\ s1t5,\ s2t5,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t1 + s1t5 < s0t2 + s2t5$
$\land\ s0t1 + s1t5 < s0t3 + s3t5$
$\land\ s0t1 + s1t5 < s0t4 + s4t5$
$\land\ s0t1 + s0t2 < s1t5 + s2t5$
$\land\ s0t1 + s0t3 < s1t5 + s3t5$
$\land\ s1t5 + s4t5 < s0t1 + s0t4$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
$\land\ s0t3 + s0t4 < s3t5 + s4t5$
BY $Z3$

THEOREM $Cx2 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s0t4,\ s1t5,\ s2t5,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t1 + s1t5 < s0t2 + s2t5$
$\land\ s0t1 + s1t5 < s0t3 + s3t5$
$\land\ s0t1 + s1t5 < s0t4 + s4t5$
$\land\ s1t5 + s2t5 < s0t1 + s0t2$
$\land\ s0t1 + s0t3 < s1t5 + s3t5$
$\land\ s1t5 + s4t5 < s0t1 + s0t4$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
$\land\ s0t3 + s0t4 < s3t5 + s4t5$
BY $Z3$

THEOREM $Cx3 \triangleq$
$\neg\exists\, s0t2,\ s0t3,\ s0t4,\ s0t5,\ s1t2,\ s1t3,\ s1t4,\ s1t5 \in Nat :$
$\land\ s0t2 + s0t3 < s1t2 + s1t3$
$\land\ s0t2 + s0t4 < s1t2 + s1t4$
$\land\ s1t2 + s1t5 < s0t2 + s0t5$
$\land\ s1t3 + s1t4 < s0t3 + s0t4$
$\land\ s0t3 + s0t5 < s1t3 + s1t5$
$\land\ s0t4 + s0t5 < s1t4 + s1t5$
BY $Z3$

THEOREM $Cx4 \triangleq$
$\neg\exists\, s0t2,\ s0t3,\ s0t4,\ s0t5,\ s1t2,\ s1t3,\ s1t4,\ s1t5 \in Nat :$
$\land\ s1t2 + s1t3 < s0t2 + s0t3$
$\land\ s0t2 + s0t4 < s1t2 + s1t4$
$\land\ s1t2 + s1t5 < s0t2 + s0t5$
$\land\ s1t3 + s1t4 < s0t3 + s0t4$

$\land\ s0t3 + s0t5 < s1t3 + s1t5$
$\land\ s0t4 + s0t5 < s1t4 + s1t5$
BY $Z3$

THEOREM $Cx5 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s0t4,\ s1t5,\ s2t5,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t1 + s1t5 < s0t2 + s2t5$
$\land\ s0t1 + s1t5 < s0t3 + s3t5$
$\land\ s0t1 + s1t5 < s0t4 + s4t5$
$\land\ s0t1 + s0t3 < s1t5 + s3t5$
$\land\ s1t5 + s4t5 < s0t1 + s0t4$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
$\land\ s0t3 + s0t4 < s3t5 + s4t5$
BY $Z3$

THEOREM $Cx6 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s0t4,\ s1t5,\ s2t5,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t3 + s3t5 < s0t1 + s1t5$
$\land\ s0t3 + s3t5 < s0t2 + s2t5$
$\land\ s0t3 + s3t5 < s0t4 + s4t5$
$\land\ s0t1 + s0t3 < s1t5 + s3t5$
$\land\ s1t5 + s4t5 < s0t1 + s0t4$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
$\land\ s0t3 + s0t4 < s3t5 + s4t5$
BY $Z3$

THEOREM $Cx7 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s1t2,\ s1t4,\ s2t5,\ s3t4,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t1 + s1t4 < s0t3 + s3t4$
$\land\ s0t3 + s3t5 < s0t2 + s2t5$
$\land\ s1t4 + s3t4 < s0t1 + s0t3$
$\land\ s1t2 + s2t5 < s1t4 + s4t5$
$\land\ s0t2 + s0t3 < s2t5 + s3t5$
$\land\ s2t5 + s4t5 < s1t2 + s1t4$
BY $Z3$

THEOREM $Cx8 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s1t4,\ s1t5,\ s2t4,\ s2t5,\ s3t4,\ s3t5 \in Nat :$
$\land\ s0t2 + s2t4 < s0t1 + s1t4$
$\land\ s0t2 + s2t4 < s0t3 + s3t4$
$\land\ s0t1 + s1t5 < s0t2 + s2t5$
$\land\ s0t1 + s1t5 < s0t3 + s3t5$
$\land\ s0t1 + s0t2 < s1t4 + s2t4$
$\land\ s0t1 + s0t2 < s1t5 + s2t5$
$\land\ s0t1 + s0t3 < s1t4 + s3t4$

$\land\ s0t1 + s0t3 < s1t5 + s3t5$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s2t5 + s3t5 < s2t4 + s3t4$
$\land\ s1t4 + s1t5 < s2t4 + s2t5$
$\land\ s1t4 + s1t5 < s3t4 + s3t5$
BY $Z3$

THEOREM $Cx9 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s1t4,\ s1t5,\ s2t4,\ s2t5,\ s3t4,\ s3t5 \in Nat :$
$\land\ s0t2 + s2t4 < s0t1 + s1t4$
$\land\ s0t2 + s2t4 < s0t3 + s3t4$
$\land\ s0t1 + s1t5 < s0t2 + s2t5$
$\land\ s0t1 + s1t5 < s0t3 + s3t5$
$\land\ s1t4 + s2t4 < s0t1 + s0t2$
$\land\ s1t4 + s2t4 < s1t5 + s2t5$
$\land\ s0t1 + s0t3 < s1t4 + s3t4$
$\land\ s0t1 + s0t3 < s1t5 + s3t5$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s2t5 + s3t5 < s2t4 + s3t4$
$\land\ s1t4 + s1t5 < s2t4 + s2t5$
$\land\ s1t4 + s1t5 < s3t4 + s3t5$
BY $Z3$

THEOREM $Cx10 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s1t4,\ s1t5,\ s2t4,\ s2t5,\ s3t4,\ s3t5 \in Nat :$
$\land\ s0t3 + s3t4 < s0t1 + s1t4$
$\land\ s0t3 + s3t4 < s0t2 + s2t4$
$\land\ s0t2 + s2t5 < s0t1 + s1t5$
$\land\ s0t2 + s2t5 < s0t3 + s3t5$
$\land\ s1t4 + s2t4 < s0t1 + s0t2$
$\land\ s1t4 + s2t4 < s1t5 + s2t5$
$\land\ s1t5 + s3t5 < s0t1 + s0t3$
$\land\ s1t5 + s3t5 < s1t4 + s3t4$
$\land\ s0t2 + s0t3 < s2t4 + s3t4$
$\land\ s0t2 + s0t3 < s2t5 + s3t5$
$\land\ s1t4 + s1t5 < s2t4 + s2t5$
$\land\ s1t4 + s1t5 < s3t4 + s3t5$
BY $Z3$

THEOREM $Cx11 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s1t4,\ s1t5,\ s2t4,\ s2t5,\ s3t4,\ s3t5 \in Nat :$
$\land\ s0t3 + s3t4 < s0t1 + s1t4$
$\land\ s0t3 + s3t4 < s0t2 + s2t4$
$\land\ s0t2 + s2t5 < s0t1 + s1t5$
$\land\ s0t2 + s2t5 < s0t3 + s3t5$
$\land\ s0t1 + s0t2 < s1t4 + s2t4$
$\land\ s0t1 + s0t2 < s1t5 + s2t5$

$\land\ s1t5 + s3t5 < s0t1 + s0t3$
$\land\ s1t5 + s3t5 < s1t4 + s3t4$
$\land\ s2t4 + s3t4 < s0t2 + s0t3$
$\land\ s2t4 + s3t4 < s2t5 + s3t5$
$\land\ s1t4 + s1t5 < s2t4 + s2t5$
$\land\ s1t4 + s1t5 < s3t4 + s3t5$
BY $Z3$

THEOREM $Cx12 \triangleq$
$\neg\exists\, s0t2,\ s0t3,\ s0t4,\ s0t5,\ s1t2,\ s1t3,\ s1t4,\ s1t5 \in Nat :$
$\land\ s0t2 + s0t4 < s1t2 + s1t4$
$\land\ s1t2 + s1t5 < s0t2 + s0t5$
$\land\ s1t3 + s1t4 < s0t3 + s0t4$
$\land\ s0t3 + s0t5 < s1t3 + s1t5$
$\land\ s0t4 + s0t5 < s1t4 + s1t5$
BY $Z3$

THEOREM $Cx13 \triangleq$
$\neg\exists\, s0t2,\ s0t3,\ s0t4,\ s1t2,\ s1t3,\ s1t4,\ s2t5,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t3 + s3t5 < s0t2 + s2t5$
$\land\ s0t3 + s3t5 < s0t4 + s4t5$
$\land\ s1t2 + s2t5 < s1t3 + s3t5$
$\land\ s1t2 + s2t5 < s1t4 + s4t5$
$\land\ s0t2 + s0t3 < s1t2 + s1t3$
$\land\ s0t2 + s0t3 < s2t5 + s3t5$
$\land\ s0t2 + s0t4 < s1t2 + s1t4$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
$\land\ s1t3 + s1t4 < s0t3 + s0t4$
$\land\ s1t3 + s1t4 < s3t5 + s4t5$
BY $Z3$

THEOREM $Cx14 \triangleq$
$\neg\exists\, s0t2,\ s0t3,\ s0t4,\ s1t2,\ s1t3,\ s1t4,\ s2t5,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t3 + s3t5 < s0t2 + s2t5$
$\land\ s0t3 + s3t5 < s0t4 + s4t5$
$\land\ s1t2 + s2t5 < s1t3 + s3t5$
$\land\ s1t2 + s2t5 < s1t4 + s4t5$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s2t5 + s3t5 < s1t2 + s1t3$
$\land\ s0t2 + s0t4 < s1t2 + s1t4$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
$\land\ s1t3 + s1t4 < s0t3 + s0t4$
$\land\ s1t3 + s1t4 < s3t5 + s4t5$
BY $Z3$

THEOREM $Cx15 \triangleq$
$\neg\exists\, s0t1,\ s0t2,\ s0t3,\ s0t4,\ s1t5,\ s2t5,\ s3t5,\ s4t5 \in Nat :$

$\land\ s0t1 + s1t5 < s0t2 + s2t5$
$\land\ s0t1 + s1t5 < s0t3 + s3t5$
$\land\ s0t1 + s1t5 < s0t4 + s4t5$
$\land\ s0t1 + s0t3 < s1t5 + s3t5$
$\land\ s1t5 + s4t5 < s0t1 + s0t4$
$\land\ s2t5 + s3t5 < s0t2 + s0t3$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
BY $Z3$

THEOREM $Cx16 \triangleq$
$\neg\exists\, s0t2,\ s0t3,\ s0t4,\ s0t5,\ s1t2,\ s1t3,\ s1t4,\ s1t5 \in Nat :$
$\land\ s0t2 + s0t4 < s1t2 + s1t4$
$\land\ s1t2 + s1t5 < s0t2 + s0t5$
$\land\ s1t3 + s1t4 < s0t3 + s0t4$
$\land\ s0t3 + s0t5 < s1t3 + s1t5$
BY $Z3$

THEOREM $Cx17 \triangleq$
$\neg\exists\, s0t2,\ s0t3,\ s0t4,\ s1t2,\ s1t3,\ s1t4,\ s2t5,\ s3t5,\ s4t5 \in Nat :$
$\land\ s0t3 + s3t5 < s0t2 + s2t5$
$\land\ s0t3 + s3t5 < s0t4 + s4t5$
$\land\ s1t2 + s2t5 < s1t3 + s3t5$
$\land\ s1t2 + s2t5 < s1t4 + s4t5$
$\land\ s0t2 + s0t4 < s1t2 + s1t4$
$\land\ s0t2 + s0t4 < s2t5 + s4t5$
$\land\ s1t3 + s1t4 < s0t3 + s0t4$
$\land\ s1t3 + s1t4 < s3t5 + s4t5$
BY $Z3$