Computation and design of locally-connected Internet exchange networks

Michael Chang and Mikhail Khodak
18 January 2016

1 Introduction

The Internet is comprised of independently operated networks (Internet Service Providers, or ISPs) that work to bring traffic from content providers to end users. To facilitate the handoff of traffic, ISPs commonly form agreements with each other. From the perspective of the end user, it is critical that their Internet content loads quickly; while there are several factors impacting the page load time, one factor is the distance travelled and the number of hops between the content providers and the end users. In first world countries like the United States, these values are low since since content providers and ISPs frequently form direct peering agreements between each other to help reduce the number of hops. However, in third world countries such as Kenya, a phenomenon known as tromboning occurs. Tromboning is the process where ISPs decide to use international connections to facilitate the domestic traffic exchange; the end users suffers because of the longer delivery time, but is less expensive to the ISP then a direct domestic connection.

In most developed countries, Internet Exchange Points (or IXPs), are used to simplify domestic peering. An example of an IXP is shown in Figure 1. Instead of a direct link between ISP 1 and ISP 2, they each connect to the IXP which will forward traffic between the two ISPs. The benefit of this is evident when there are many ISPs that would benefit from domestic interconnection; an ISP would only need a single link to the IXP in order to connect to all the other ISPs (that are already linked to the IXP). Naturally, IXPs are subject to network effects; the more ISPs that partner up, the more effective and valuable the IXP. However, it is an arduous and potentially fruitless effort to recruit this critical mass of ISPs into the connection without some kind of incentive.

In this project, we construct a profit model that describes two of the equilibrium states of a local Internet system - one in which all ISPs route traffic to each other internationally or all ISPs route traffic to each other through an IXP. We formulate an incentive scheme whose cost can be minimized via a quadratically-constrained integer program and consider approaches to solving it. Furthermore, we consider a separate model under which ISPs and content provider form a network which whose construction and routing costs we wish to minimize. We evaluate the performance of algorithms we consider on real-world and simulated data.

For all fully implemented algorithms, data sets, and scrapers, please visit our Github: $https://github.com/mchang6137/IXP\_incentives$

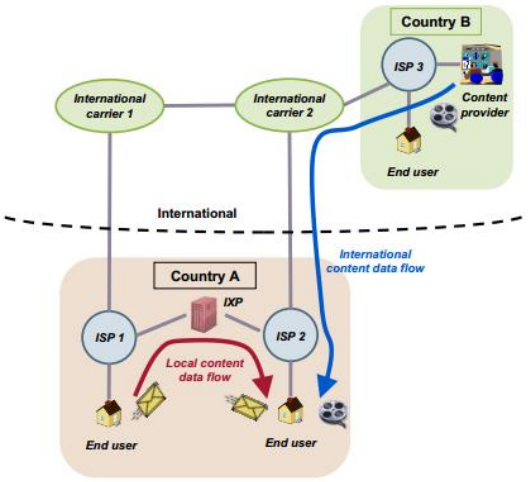

Figure 1: Insertion of an IXP to avoid the "tromboning" effect

2 Computing ISP incentives for local IXP connection

Under fairly general assumptions, it is straightforward to construct an ISP profit model where the optimal state, i.e. the state in which all ISPs are connected to a local IXP, is a Nash equilibrium. However, since constructing and connecting to an IXP is costly and yields no benefit to a single ISP without multilateral actions from other ISPs, the state in which all ISPs route traffic through an international exchange is also a Nash equilibrium. In this section, we use optimization theory to compute incentives so as to incur the least amount of cost while incentivizing ISPs to connect to a local IXP, i.e. to transition from the poor Nash equilibrium to the optimal one. We first construct the profit model, then present the optimization problem and methods for solving it, and finally discuss results on real and hypothetical data.

2.1 ISP Profit Model

We define the following variables:

- V - set of ISPs
- S - subset of ISPs in V that are connected to the local IXP
- w_{ij} - traffic from node i to node j
- p_{IXP} price charged by local IXP
- p_{INT} price charged by international IXP
- r - interest rate
- C_i - cost to connect ISP i to the local IXP

Using $.95 \cdot \max\{w_{ij}, w_{ji}\}$ as the transit charge, we then have the following ISP time-discounted utility function at some initial time $t = 1$ for an ISP $i \in V \backslash S$:

$$u_i = .95 \cdot \sum_{t=0}^{\infty} r^t p_{INT} \sum_{j \in V} \max\{w_{ij}, w_{ji}\}$$

If the ISP decides to switch to the IXP at time $t = 0$, we instead have the following utility function:

$$\tilde{u}_i = .95 \cdot \sum_{t=0}^{\infty} r^t \left(p_{IXP} \sum_{j \in S} \max\{w_{ij}, w_{ji}\} + p_{INT} \sum_{j \in V \backslash S} \max\{w_{ij}, w_{ji}\} \right) - C_i$$

Setting $Z = \sum_{t=0}^{\infty} (p_{INT} - p_{IXP}) r^t = \frac{p_{INT} - p_{IXP}}{1-r}$ and $M_{ij} = .95 \cdot \max\{w_{ij}, w_{ji}\}$, we have that the net benefit to ISP i from switching to the IXP is

$$\Delta u_i = \tilde{u}_i - u_i = Z \sum_{j \in S} M_{ij} - C_i$$

Assuming $w_{ii} = 0$ and $C_i > 0 \ \forall \ i \in V$, if no ISP has connected then $\Delta u_i = Z \sum_{j \in S} M_{ij} - C_i = -C_i < 0 \ \forall \ i$, so no ISP will have incentive to unilaterally switch and therefore the state is a Nash equilibrium.

2.2 Incentive Computation Problem

Suppose we are planning to build an IXP and wish to incentivize ISPs in the country to join. Under the profit model we have just devised, the way to do this would be to pay for several ISPs to connect and have the rest join due to a network effect. We would like to pay for as few ISPs as possible to connect, so we should find a subset of ISPs minimizing the sum of connection costs over that set. The constraint will then be that $\Delta u_i \geq 0$ for all ISPs i not in the incentivized subset. As an optimization problem, this can be written as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in S} C_i \\ \text{subject to} \quad & Z \sum_{j \in S} M_{ij} - C_i \geq 0 \ \ \forall \ \ i \in V \backslash S \\ & S \in 2^V \end{aligned}$$

This can be rewritten as the following combinatorial optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{|V|} C_i x_i \\ \text{subject to} \quad & \left(Z \sum_{j=1}^{|V|} M_{ij} x_j - C_i \right) (1 - x_i) \geq 0 \ \ \forall \ \ i \in V \\ & x \in \{0,1\}^{|V|} \end{aligned}$$

We can then relax the problem to a quadratically-constrained linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{|V|} C_i x_i \\ \text{subject to} \quad & Z \sum_{j=1}^{|V|} M_{ij} x_j - Z \sum_{j=1}^{|V|} M_{ij} x_i x_j + C_i x_i - C_i \geq 0 \ \ \forall \ \ i \in V \\ & 0 \leq x_i \leq 1 \ \ \forall \ \ i \in V \end{aligned}$$

QCLPs have standard form:

$$\begin{aligned} \text{minimize} \quad & q_0^T x \\ \text{subject to} \quad & x^T P_i x + q_i^T x + r_i \leq 0 \ \ \forall \ \ i \in V \\ & 0 \leq x_i \leq 1 \ \ \forall \ \ i \in V \end{aligned}$$

In order to set our problem into this form, we need to add each quadratic inequality constraint to its own tranpose so that the matrices P_i are symmetric. Our problem then has

$$q_0 = C \quad P_i = Z\left(e_i M_i^T + M_i e_i^T\right) \quad q_i = -2(ZM_i + C_i e_i) \quad r_i = 2C_i$$

QCLP problems are NP-hard in general but convex for $P_i \succeq 0 \ \forall \ i$. Since the matrix P_i is not positive semi-definite in general, this problem is nonconvex. We must therefore look at nonconvex optimization techniques. Since QCLPs are a special case of the more general and also more commonly encountered quadratically-constrained quadratic programs (QCQPs), many of our methods will be drawn from methods for such problems.

2.3 Semidefinite relaxation

The most straightforward way of solving QCLPs is by using the following SDP relaxation [1, p. 3]:

$$\begin{aligned} \text{minimize} \quad & q_0^T x \\ \text{subject to} \quad & \text{tr}(P_i X) + q_i^T x + r_i \leq 0 \ \ \forall \ \ i \in V \\ & 0 \leq x_i \leq 1 \ \ \forall \ \ i \in V \\ & X - xx^T \succeq 0 \end{aligned}$$

The relaxation here is converting the nonconvex constraint $X - xx^T = 0$ to $X - xx^T \succeq 0$, which can be reformulated as a Schur complement:

$$\begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0$$

In fact, solving this SDP is effectively solving the bidual of the original QCLP. The relaxation has $O(n)$ constraints and variables so for our case computing the optimal value is straightfoward.

2.4 Iterative linearization

We can linearize the QCLP problem by using the decomposition $P_i = P_i^+ - P_i^-$ s.t. $P_i^+, P_i^- \succeq 0$, which is easily done by conducting a spectral decomposition of P and then taking separate sums over positive and negative eigenvalues. Then we can transform constraint $x^T P_i x + q_i^T x + r_i \leq 0$ into $x^T P_i^+ x + q_i^T x + r_i \leq x^T P_i^- x$ and linearize around some point x_0 to get

$$x^T P_i^+ x + q_i^T x + r_i \leq x_0^T P_i^- x_0 + 2x_0^T P_i^- (x - x_0)$$

This is now a convex constraint that, if initialized at a suitable point x_0 can be solved to yield a new feasible point x_1, and we can continue to solve this problem iteratively [2, p. 9]. Although finding a feasible point x_0 can be hard for some problems, in our case the all-ones vector works, as if it is infeasible then the problem itself is infeasible. This program has $O(n)$ constraints and variables and can also be computed for our cases.

2.5 Reformulation-linearization technique

We first rewrite the QCLP as

$$\begin{aligned} \text{minimize} \quad & q_0^T x \\ \text{subject to} \quad & \sum_{j=1}^{|V|}\sum_{j=1}^{|V|} P_{i,jk} x_j x_k + q_i^T x_i + r_i \leq 0 \ \ \forall \ \ i \in V \\ & l_i \leq x_i \leq u_i \ \ \forall \ \ i \in V \end{aligned}$$

where in our case $l_i = 0, u_i = 1 \ \forall \ i$. Since this last constraint on x_i implies

$$(u_i - x_i)(u_j - x_j) \geq 0, \quad (u_i - x_i)(x_j - l_j) \geq 0, \quad (x_i - l_i)(u_j - x_j) \geq 0, \quad (x_i - l_i)(x_j - l_j) \geq 0$$

which then imply

$$x_i x_j \geq u_j x_i + u_i x_j - u_i u_j, \quad x_i x_j \leq l_j x_i + u_i x_j - u_i l_j, \quad x_i x_j \leq u_j x_i + l_i x_j - l_i u_j, \quad x_i x_j \geq l_j x_i + l_i x_j - l_i l_j$$

by substituting $X_{ij} = x_i x_j$ we can linearly relax this QCLP into the following LP [7, p. 269]:

$$\begin{aligned} \text{minimize} \quad & q_0^T x \\ \text{subject to} \quad & \sum_{j=1}^{|V|}\sum_{j=1}^{|V|} P_{i,jk} X_{jk} + q_i^T x_i + r_i \leq 0 \ \ \forall \ \ i \in V \\ & X_{ij} - u_j x_i - u_i x_j \geq -u_i u_j \ \ \forall \ \ i \in V \\ & X_{ij} - l_j x_i - u_i x_j \geq -u_i l_j \ \ \forall \ \ i \in V \\ & X_{ij} - u_j x_i - l_i x_j \geq -l_i u_j \ \ \forall \ \ i \in V \\ & X_{ij} - l_j x_i - l_i x_j \geq -l_i l_j \ \ \forall \ \ i \in V \end{aligned}$$

Unlike the nonlinear relaxations, this LP reformulation has $O(n^2)$ variables and $O(n^2)$ constraints, which while solvable for our case is much more intensive in both memory and compute-time.

2.6 Simulated annealing

Since in the unrelaxed minimization problem over sets S in the powerset of V it is easy to find a feasible starting point (set $S = V$) and straightforward to devise sampling rules, the original optimization lends itself well to an approach via simulated annealing. We use the following algorithm for the target function $f(S) = \sum_{i \in S} C_i$ with feasibility constraints $Z \sum_{j \in S} M_{ij} - C_i \geq 0 \; \forall \; i \in V \backslash S$:

1 : set $S_0 = S$

2 : for $t \in \{1, \ldots, |V|^2\}$:

3 : set $\tilde{S} = S_{t-1}$

4 : for $i \in V$:

5 : add i to $\tilde{S}$ if $i \in V \backslash \tilde{S}$ or remove i from $\tilde{S}$ if $i \in \tilde{S}$ with probability $\dfrac{1}{2(\log_{10}(t) + 1)}$

6 : if $\tilde{S}$ is feasible:

7 : repeatedly remove elements $\arg \max_{\substack{i \in \tilde{S} \\ \tilde{S} \backslash \{i\} \text{ is feasible}}} C_i$ from $\tilde{S}$

8 : set $S_t = \tilde{S}$ with probability $\min \left\{ 1, \dfrac{f(S_{t-1})}{f(\tilde{S})} \right\}$

9 : keep track of the best feasible set seen so far

10 : return best feasible set that was seen

Note that in step 7 of this algorithm we repeatedly greedily remove the largest feasible element from S since our objective is strictly decreasing with fewer elements in the set.

2.7 Rounding methodology

Apart from simulated annealing, the all solution methods discussed yield values $x_i \in [0, 1] \; \forall \; i \in V$, which we then need to round to $x_i \in \{0, 1\}$ in order to assign ISP i to S or $V \backslash S$. We present two methods of doing so: one deterministic and one randomized. For deterministic rounding, we have the following algorithm:

1 : given vector $x \in [0, 1]^{|V|}$

2 : set $S = \emptyset$

3 : while S is infeasible:

4 : set $k = \arg \max_{i \in V} x_i$

5 : set $x_k = 0, S = S \cup \{k\}$

6 : repeatedly remove elements $\arg \max_{\substack{i \in S \\ S \backslash \{i\} \text{ is feasible}}} C_i$ from S

7 : return S

The method for randomized rounding is given below:

1 : given vector $x \in [0,1]^{|V|}$

2 : for $t = \{1, \ldots, |V|^2\}$:

3 : set $S_t = \emptyset$

4 : while S_t is infeasible:

5 : randomly draw an index $k \in V$ under the distribution $\frac{x}{\|x\|_1}$ over V

6 : set $x_k = 0, S_t = S_t \cup \{k\}$

7 : repeatedly remove elements $\arg \max_{\substack{i \in S_t \\ S_t \setminus \{i\} \text{ is feasible}}} C_i$ from S_t

8 : return $\arg\min_t \sum_{i \in S_t} C_i$

Note that, as with simulated annealing, at step 6 in the deterministic rounding and step 7 in the randomized algorithm we greedily "tighten" our solution by repeatedly removing the largest feasible element from the set, bringing the solution closer to the constraint while maintaining feasibility.

3 Incentive computation results

We test the different ways of computing incentives on both approximately realistic and self generated data. Given a set of ISP user weights $W_i \ \forall \ i \in V$, we calculate ISP i to j Internet traffic weight w_{ij} using the formula

$$w_{ij} = \frac{W_i}{\sum_{k \neq j} W_k} \cdot 1_{i \neq j}$$

which sets $w_{ii} = 0 \ \forall \ i \in V$. W_{ij} models the volume of traffic that is being sent from ISP i to ISP j. If ISP i house many destination prefixes, we can expect that it will have more outgoing traffic as these servers need to respond to a high quantity of user queries. On the other hand, the number of destination prefixes is irrelevant to how many requests that ISP i is making, so we exclude that value when measuring W_{ij} in relation to the amount of traffic at large flowing through our local system. For costs, since connecting an ISP that sends a lot of traffic is more costly than connecting a smaller ISP, cost should be increasing with increasing weight W_i. For our tests, we use $C_i = \frac{\ln(W_i)+1}{\bar{W}}$, where $\bar{W}$ is the average ISP weight. For the constants, we use transit prices $p_{INT} = 1.1$ and $p_{IXP} = 1.2$ and interest rate $r = .05$.

3.1 Simulation results

For $|V| = \{5, 10, 20, 40, 80\}$ we generate weights W_i by setting $\frac{3}{5}$ths of the weights between 1 and 100, $\frac{1}{5}$th of the weights between 100 and 500, and $\frac{1}{5}$th of the weights between 500 and 2000, drawing uniformly at random on those intervals. The resulting weight distributions reflect values seen in real-world networks. We conduct all simulations using `MATLAB CVX`.

$\|V\| = 40$	OPT	max round	rand round	exp round	iter	time
SDR	0.00	.6102	.5752	.5795	-	1.9609
Iter. Lin.	.2405	.5752	.5752	.5752	-	27.3188
RLT	.0663	.5752	.5752	.5756	1.9	9.1641
Sim. anneal.	.5660	-	-	-	537	.0234

The results for $|V| = 40$ are summarized in the previous table. As can be seen, the simulated does best in terms of both runtime and absoute value. Among the relaxations, the iterative linearization and the reformulation perform similarly, although the former takes longer to run. Note that although the relaxation methods have the same randomized rounding results, the expected value (for one trial of randomized rounding) is different. Looking at the optimum value for SDP (which is negligible but nonzero), we note that it is likely over-relaxed for this problem.

3.2 Simulation on real-world data

As stated earlier, the weight on each ISP is modeled by the number of destination prefixes advertised from that ISP. To do this, we accessed data from the Routeviews Project. We examined Routing Information Base (RIB) tables, which stored information from the Border Gateway Protocol (BGP). In particular, we looked at the data from Kenya IXP (KIXP), London IXP (LINX), and Johannesburg IXP (JIXP). We performed our analysis on the basis of examining AS Paths, and matching up the last autonomous system (AS) on that path and counted the number of destination prefixes advertised. We discounted autonomous systems that were not publicly known to be a direct peer of the IXP, as it is atypical to have destination prefixes advertised from autonomous systems that are not peering with the IXP. We note that this approach has major limitations; for example, one prefix (such as Google) might experience significantly more traffic then a lesser accessed prefix. For future work, we will look into collecting better data and speaking to ISP operators to develop a more nuanced understanding of the traffic flow between the ISP members of an IXP.

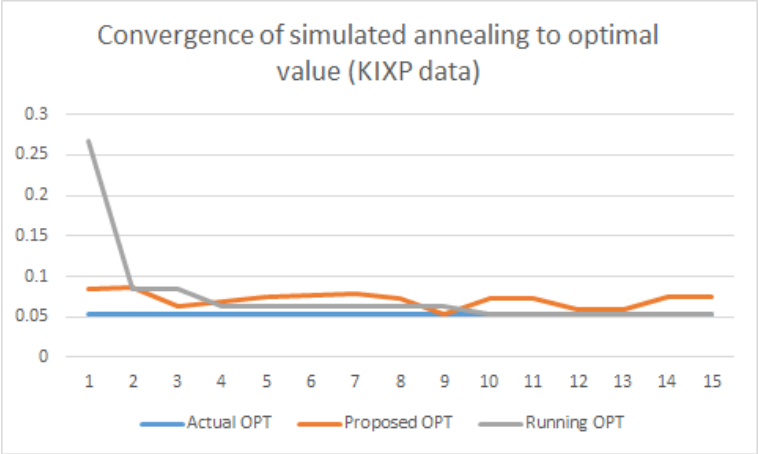

Figure 2: Simulated Annealing on $|V| = 23$

The table below shows incentive computation results for the KIXP, which has the following 23 node weights

$$\{4234, 1603, 9, 5, 1, 28, 1, 1, 59, 17, 9, 1, 81, 288, 1607, 2, 1, 13, 139, 90, 78, 164, 35\}$$

The optimal solution, found through brute force over $2^2 3 - 1$ sets, was to incentivize the first ISP (by far the largest) as well a few smaller ones. This reflects the fact that the largest ISP will have an outside influence on other ISPs joining the IXP. The computational results are summarized below. As before, simulated annealing performed best, often converging very quickly to the result (Figure 2).

KIXP	OPT	max round	rand round	exp round	iter	time
Brute force	.0528	-	-	-	-	51.64
SDR	0.00	.1610	.0528	.0991	-	.4844
Iter. Lin.	.0385	.0594	.0528	.0637	-	6.2031
RLT	.0247	.0603	.0528	.0603	5	5.6250
Sim. anneal.	.0528	-	-	-	53	.1406

4 Minimizing local network setup and transit cost

In the previous section, we considered the situation in which the aim is to setup a direct connection between every ISP and a local IXP in order to shorten the transit route between local service providers. Since much international traffic and even local traffic in developing countries is due to large content providers (e.g. Google, Facebook), who themselves have a large incentive to increase Internet penetration because it increases their user base, an alternative to establishing a separate IXP is to set up data centers for these large companies at one of the local ISPs and have the rest of the ISPs connect to them via peering agreements. We then would like to minimize the cost of setting up and routing traffic through this network. In this section we consider how to best model this situation and present several algorithms for computing an optimal network, along with numerical results on real and hypothetical data.

4.1 Related work

We wish to construct a network with a minimal combined construction and operation cost. One potential way to model this situation would be by considering a connected facilities location problem, in which given a graph $G(V, E)$ with edge costs c_e and a set of demands $D \subset V$, we wish to establish a set of facilities $F \subset V$ minimizing

$$\sum_{i \in D} w_i \sum_{e \in P_{i,F}} c_e + M \sum_{e \in T_F} c_e$$

where $P_{i,F}$ is the shortest path from i to F, T_F is a Steiner tree on F, and $M > 1$ is a parameter. For this problem, which is also known as a single-sink rent-or-buy problem, there exists randomized approximation scheme to compute F and cost-sharing mechanism that ensures fairness and that subsets of agents do not

defect to form their own networks [4, p. 2-3][5, p. 141]. However, in our case this game would effectively be establishing a backbone network between facilities F, which are given unlimited routing for an additional cost M, an unrealistic scenario for our purposes. At best we could use an extension of the model in which establishing a facility would only be allowed at one node, but in this case the algorithm proposed by the authors essentially just creates a minimum spanning tree. From a game-theoretic perspective, a very-much related game in which each ISP $i \in V$ sets up links to other ISPs, selfishly minimizing the sum of the cost of setting up the links as well as the distance between itself and other ISPs, was proposed by Fabrikant et al. [3]. Our model is in many ways similar to their game.

4.2 Network design model

We define the following variables:

- V - set of ISPs
- w_{ij} - traffic from node i to node j
- p - price per unit of routed traffic along an edge
- c_e - cost of constructing an edge $e \in V \times V$
- r - interest rate

Using $.95 \cdot \max\{w_{ij}, w_{ji}\}$ as the transit charge, we then have the following time-discounted total cost function $f(E)$ for any set of edges $E \subset V \times V$:

$$f(E) = .95 \cdot \sum_{t=1}^{\infty} r^t \sum_{(i,j) \in V \times V} \sum_{e \in P_{ij}^{(E)}} p \max\{w_{ij}, w_{ji}\} + \sum_{e \in E} c_e = Z \sum_{(i,j) \in V \times V} |P_{ij}^{(E)}| M_{ij} + \sum_{e \in E} c_e$$

where $P_{ij}^{(E)}$ is the set of edges along the shortest path in $G(V, E)$ between i and j, $Z = \sum_{t=0}^{\infty} pr^t = \frac{p}{1-r}$, and $M_{ij} = .95 \cdot \max\{w_{ij}, w_{ji}\}$. In effect this function models the total cost of routing traffic between each node, added to the cost of constructing edges. Since constructing more edges reduces the shortest-path length between nodes, the solution graph $G(V, E)$ does not necessarily have to be a tree. The function is highly impractical to minimize using non-heuristic methods, with connectivity constraints and shortest-path computations making it difficult to relax into a non-Boolean optimization problem. Furthermore, we make no metric assumption on the weights c_e, further complicating the situation. We will instead consider two greedy heuristic methods as well as simulated annealing in our approach to this problem. All implementations below done in `Python` using the `networkx` module.

4.3 Greedy algorithms

We construct two greedy algorithms. Denoting as before $P_{ij}^{(E)}$ to be the set of edges along the shortest path in $G(V, E)$ between i and j, T to be the set of

edges in the minimum-cost spanning tree on nodes V with edge-weights c_e, and K to be the set of edges in the complete graph, the first method is as follows:

1 : set $E = T$

2 : while True:

3 : set $a = \arg\min_{(i,j)\in V\times V} f\left(E \cup P_{ij}^{(K)}\right) - f(E)$

4 : if $f\left(E \cup P_{a}^{(K)}\right) - f(E) < 0$: $E = E \cup P_{a}^{(K)}$

5 : set $b = \arg\min_{e\in E} f(E\backslash\{e\}) - f(E)$

6 : if $f(E\backslash\{b\}) - f(E) < 0$: $E = E\backslash\{b\}$

7 : break if no change from last loop

9 : return E

This algorithm works by greedily adding a shortest path if doing so improves the optimal value and then greedily removing an edge if doing so improves the optimal value.

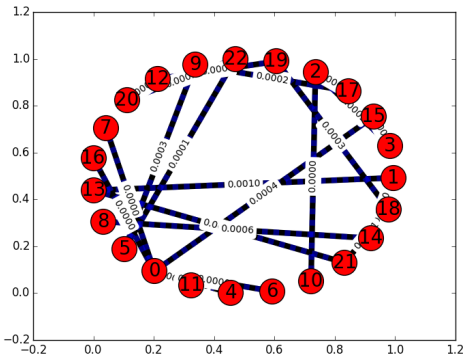

Figure 3: For the implementation of all our algorithms, we started this this Minimum Spanning Tree constructed using Kruskal's algorithm

The second greedy algorithm works the same way but instead starts with a complete graph:

1 : set $E = K$

2 : while True:

3 : set $a = \arg\min_{e\in E} f(E\backslash\{e\}) - f(E)$

4 : if $f(E\backslash\{a\}) - f(E) < 0$: $E = E\backslash\{a\}$

5 : set $b = \arg\min_{(i,j)\in V\times V} f\left(E \cup P_{ij}^{(K)}\right) - f(E)$

6 : if $f\left(E \cup P_{b}^{(K)}\right) - f(E) < 0$: $E = E \cup P_{b}^{(K)}$

7 : break if no change from last loop

9 : return E

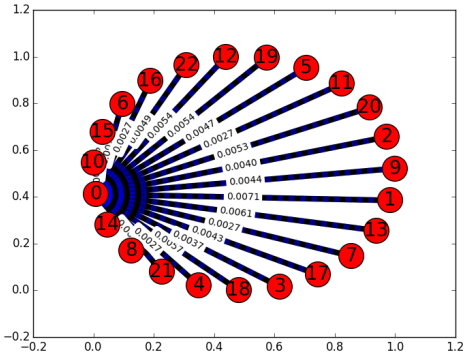

Figure 4: After running the first greedy algorithm, this was the resulting graph

Naive implementations of both greedy algorithms are time-intensive due to the necessity of computing shortest paths and taking minima over them, which since the graph may be complete takes $\mathcal{O}(|V|^4)$ time. However, in practice this worst-case single-loop time does not hold since the shortest path between two nodes is often simply the edge connecting them, and furthermore one can include optimization of maintaining an ordered list of shortest paths. Since the optimal value must decrease at each loop, under a set tolerance the algorithm will converge.

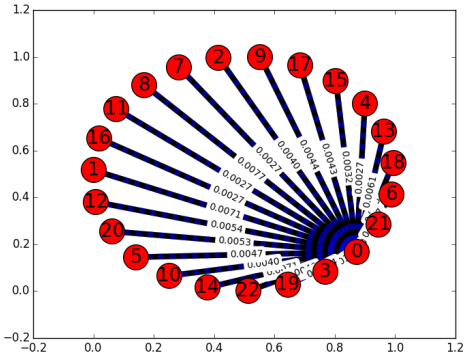

Figure 5: After running the second greedy algorithm, this was the resulting graph

4.4 Simulated annealing

We also construct a simulated annealing optimization technique for this function, sampling starting from a minimum-cost spanning tree by randomly adding shortest paths and then repeatedly removing redundant edges before comparing

with the previous graph.

1 : set E_0 to be set of edges of the minimum-cost spanning tree on nodes V with edge-weights c_e
2 : for $t \in \{1, \ldots, |V|^2\}$:
3 : set $\tilde{E} = E_{t-1}$
4 : for $(i, j) \in V \times V$:
5 : set $\tilde{E} = \tilde{E} \cup P_{ij}^{(K)}$ with probability $\dfrac{1}{2(\log_{10}(t) + 1)}$
6 : while True:
7 : set $a = \arg\min_{e \in \tilde{E}} f(\tilde{E} \backslash \{e\}) - f(\tilde{E})$
8 : if $f(\tilde{E} \backslash \{a\}) - f(\tilde{E}) < 0$: $\tilde{E} = \tilde{E} \backslash \{a\}$
9 : set $E_t = \tilde{E}$ with probability $\min \left\{ 1, \dfrac{f(E_{t-1})}{f(\tilde{E})} \right\}$
10 : keep track of the best edge set seen so far
11 : return best edge set that was seen

For simulated annealing, the minimum value of the target function was observed at the 8.349 when the number of nodes is 23. The graph below displays the result of simulated annealing on the previously described MST. A similar structure with a central node develops.

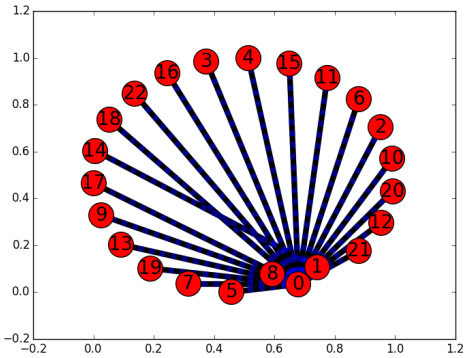

Figure 6: After running the algorithm, this was the resulting graph

5 Conclusion

In this project, we considered ways of modeling problems associated with incentives for Interent exchange points and other means of local network connectivity in developing countries. From this perspective, we found that our incentive model provided the intuitive result that it is generally best to incentivize a few of the most-heavily weighted ISPs rather than many smaller ISPs, at least under the cost function that we set. Although further work should

look at what a realistic cost function would be in this environment, our model allows a fairly straightforward interpretation of the network effect, with larger ISPs having more weight. Similarly, under some simple cost functions we found that minimum cost networks would reflect an IXP-type arrangement, with the largest ISPs serving as nodes connecting many smaller ones and thus effectively acting as IXPs. In addition to looking at the setup-cost function and other pricing details, future efforts should also consider whether more influential ISPs will charge more to be incentivized, particularly since they may be acting anti-competitively.

From a computational point of view, we examined and devised several different algorithms, both for incentive computation and for network design. We considered four different ways of approximately solving quadratically-constrained linear problems, finding that simulated annealing was by far the best algorithm in our situation, both in terms of runtime and final optimal value. Our relaxation approaches varied in terms of effectiveness, with some, such as the basic SDP relaxation, being too relaxed to be of much use. For network design, the problem we consider is quite hard and does not lend itself to many approximation or relaxation techniques. We developed two very simple greedy algorithms and compared them to simulated annealing. More work however needs to be done in developing better heuristics and guarantees for this case however.

In general, since simulated annealing has been shown to be highly succesful for both of our problems, more effort can be directed to considering better sampling and updating rules. In particular, an interesting approach would be to incorporate node weight into the probability of nodes or shortest paths between nodes being selected. For the optimization problems, better solutions such as branch-and-bound algorithms, which iteratively relax and restrict the global optimization problem to convex subsections, should be considered [7]. Another direction of future algorithmic study would be to develop cost-sharing mechanisms for sharing the cost incurred by constructing the network [6]. Although some allocative solution concepts such as the Shapley value or the nucleolus from coalitional game theory could apply, these are difficult to compute for non-tree graphs, which are a feature of the networks our algorithms found. More feasible cost sharing mechanisms could be constructed based on existing ones for the connected facilities problem, as in Gupta et al. [5].

References

[1] K. M. Anstreicher, "SDP versus RLT for Nonconvex QCQP", Workshop on Integer Programming and Continuous Optimization, Chemnitz (2004). `https://www-user.tu-chemnitz.de/~helmberg/workshop04/anstreicher.pdf`

[2] A. d'Aspremont & S. Boyd, "Relaxations and Randomized methods for nonconvex QCQPs", EE 392o. Stanford University, Palo Alto, CA. Autumn 2003. `http://stanford.edu/class/ee364b/lectures/OLDrelaxations.pdf`

[3] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, & S. Shenker, "On a network creation game", In *Proc. 22nd Annual Symp. on Principles of Distributed Computing* 2003, 347-351.

[4] A. Gupta, A. Kumar, & T. Roughgarden, "Simpler and Better Approximation Algorithms for Network Design", In *35th ACM STOC* 2003.

[5] A. Gupta, A. Srinivasan, & E. Tardos, "Cost-sharing mechanisms for network design", In *Proc. 7th APPROX* 2004, 139-150.

[6] N. Megiddo, "Computational complexity of the game theory approach to cost allocation for a tree", *Mathematics of Operations Research* **3** No. 3 1978, 189-196.

[7] T. Van Voorhis & F. Al-Khayyal, "Accelerating Convergence of Branch-and-Bound Algorithms For Quadratically Constrained Optimization Problems", *State of the Art in Global Optimization: Computational Methods and Applications*, Ed. C. A. Floudas & P. M. Pardalos, (Kluwer Academic Publishers 1996), 267-284.