GATEway: Symbiotic Inter-Domain Traffic Engineering

Matthew Roughan
University of Adelaide
SA, Australia
matthew.roughan@adelaide.edu.au

Yin Zhang
University of Texas at Austin
TX, USA
yzhang@cs.utexas.edu

ABSTRACT

There are a group of problems in networking that can most naturally be described as optimization problems (network design, traffic engineering, etc.). There has been a great deal of research devoted to solving these problems, but this research has been concentrated on intra-domain problems where one network operator has complete information and control. An emerging field is inter-domain engineering, for instance, traffic engineering between large autonomous networks. Extending intra-domain optimization techniques to inter-domain problems is often impossible without the measurements and control available within a domain.

This paper presents an alternative: we propose a method for traffic engineering that doesn't require sharing of important information across domains. The method extends the idea of genetic algorithms to allow symbiotic evolution between two parties. Both parties may improve their performance without revealing their data, other than what would be easily observed in any case. We show the method provides large reductions in network congestion, close to the optimal shortest path routing across a pair of networks. The results are highly robust to measurement noise, the method is very flexible, and it can be applied using existing routing.

1. INTRODUCTION

Global optimization is the natural approach to many problems in networking. For instance network design, traffic engineering, and routing are all optimization problems. We typically seek the solution with the global minimum cost (where cost may be an abstract measure).

However, in the Internet there is no one authority which can perform such an optimization. The Internet is broken into many Autonomous Systems (ASes), each of which is managed independently, and so any optimization must be distributed. Furthermore, these individual sub-networks are often unwilling to co-operate, and so cannot attain the global optimum. For these reasons many problems in network are treated as game-theory problems with selfish participants, each trying to optimize for their own benefit alone.

Games with selfish participants have been often studied, for example, the Prisoner's Dilemma [1]. In this game, the prisoners could maximize their joint beneficial outcome if they co-operate, but the game is such that if one prisoner acts selfishly (and confesses), he can minimize losses. If both confess, their fate is sealed. Unfortunately (for the prisoners, if not society) they cannot trust each other, and so they will not co-operate, and the mutual outcome is then the worst possible (for the prisoners).

These games were not invented for the benefit of criminals awaiting prosecution, but rather because many situations in business, and in particular in network management can be modelled the same way. Corporations are often assumed to follow a selfish model (as a result of their requirement to return benefit to their stock holders), and hence network operators are likewise assumed to follow such a model.

The key point at which we differ from the literature is that we argue that network operators are not "selfish". Rather, they are often willing to co-operate as long as they can be assured they will not be exploited. For instance current Internet routing relies on a certain amount of co-operation to ensure smooth operation. The Prisoner's Dilemma is easily resolved if both prisoners can trust each other — neither confesses and they're released. Likewise, mutually trusting network operators are often willing to co-operate to improve performance.

Attaining trust is non-trivial. Trust relationships between individuals are not sufficient, as individuals change jobs, and responsibilities. We need trust between corporate entities. This can sometimes be achieved through intense legal negotiation, but it seems unlikely that flexible, multilateral agreements can be achieved in this manner. Another approach is to have an independent third party that acts as an arbiter. Such approaches also require complex negotiation, and/or careful legislation. Again, flexibility is lacking.

This paper describes an alternative based in part on the idea of privacy-preserving distributed computation. Such computation can be used to create methods for joint optimization between networks, without the type of "risky" co-operation that most previous methods of joint optimization require. We focus here on the inter-domain Traffic Engineering (TE) problem. In particular, our method is aimed at allowing TE to proceed without the providers sharing information that they consider private. This prevents the partners in the optimization exploiting information gained about its competitors, prevents gaming of the situation, and provides a basis for trust.

We exploit two key ideas: firstly, we use an optimization heuristic based on the metaphor of Darwinian evolution, commonly called a Genetic Algorithm (GA). GAs proceed by describing the optimization variables using a series of "genes". A population is created, allowed to compete, and the most successful are allowed to reproduce. We call our approach *GATEway* (Genetic Algorithm Traffic Engineering). GAs are ideally suited to cases where the objective function is hard to compute, and we use this here by extending the metaphor to allow *symbiosis* between pairs of providers.

GameComm 2008 October 20, 2008, Athens, GREECE.

In nature, symbiotic organisms jointly evolve, but they don't need to share genetic material to do so. Analogously, GATEway allows two providers to optimize their routing without sharing the details of their own networks (their genes). Despite this secrecy, we show that on Rocketfuel networks GATEway dramatically improves performance as compared to existing provider routing, and selfish routing procedures. In fact, GATEway provides results within 5% of a reasonable lower-bound on the possible performance, and about 40% better than closest equivalent selfish routing.

As in biology, some information sharing is still required even in the above approach, primarily in the form of fitness functions. The fitness of each member of the population must be evaluated (in biology this would be implicit in whether individuals survive to breed). We then use techniques from the secure distributed computation community to substantially reduce even this modicum of information sharing. This allows the above optimization to be conducted without leaking any direct information about the providers, for instance, they do not need to share topology, link capacity, internal traffic, or routing details. In fact, in the strictest version of GATEway, the providers share almost no information at all, though there is a penalty to be paid for such parsimony. Ironically, despite sharing less information, the communication cost increases.

Applying symbiosis to GAs represents a new approach to secure distributed computation. Previously, many of the algorithms applied for secure distributed computation have been based on Yao's two-party protocol, which can compute any polynomial time function. We show here that we can find approximate solutions to NP-hard problems. The problem we consider here is quite specific, but there are many other fields where similar issues are encountered. Our approach is quite generic, and so may be applicable to other problems both in network engineering, and outside.

We further address some of the practical problems of using such a protocol. We demonstrate the flexibility of the approach by using alternative optimization objectives, and we also find that the method is highly insensitive to measurement noise. Additionally, we illustrate how this type of traffic engineering could be implemented using today's technology, demonstrating that such an approach is practical within today's networks.

2. BACKGROUND AND RELATED WORK

2.1 Traffic Engineering

There are many tasks in network operations which fall under the heading of optimization. In this paper we shall concentrate on Traffic Engineering (TE), the process of balancing one's traffic across the existing links in a network. One may think of this as optimizing the routing parameters of a network, such that the resulting routing is "beneficial" in some sense. The routing parameters determine, for each source-destination pair, the fraction of traffic going on different paths from the source to the destination. Many TE techniques have been presented (for examples see [2–15]). The majority of the TE literature concerns intra-domain TE. That is, optimization of routing parameters within a single network. There are many approaches to this problem, though the two most prevalent are given below.

Explicit path where the traffic is arbitrarily routed to satisfy the results of a multi-commodity flow optimization [16, Chapter 17]. Explicit path routing is generally instantiated through MPLS (Multi-Protocol Label Switching) or IP-in-IP encapsulation [17].

Shortest-path where the routing uses shortest-paths, but the link weights are arbitrarily chosen as the result of some optimization. Shortest path routing is appealing because it can be implemented easily using today's most commonly used Interior Gateway Protocols (IGPs) In these protocols each link is associated with a positive weight, and path length is defined as the sum of the weights of all the links on that path. Traffic is routed along the shortest paths. In cases of ties the flow is generally split roughly evenly across Multiple Equal-Cost Paths (MECP).

Explicit path optimization has less constraints, and therefore must achieve a superior solution to the shortest-path optimization. Naively, one supposes that explicit path optimization will perform significantly better. However, there is now substantial literature supporting shortest-path optimization. It has been shown that (for realistic networks) one can get within a few percent of the performance of explicit path routing [6], even where the inputs contain prediction or inference errors [13, 15]. What's more shortest-path optimization can choose sets of weights that perform well over a range of traffic (say the variations over the course of a day) [9, 15] or under link failures [14, 18, 19].

Either technique is appropriate within a single network, but both have flaws for inter-domain TE, a topic of recent interest [17, 20–28]. The Internet has a broad two-level hierarchy in its routing, separating intra-domain routing from inter-domain routing. BGPv4 (the Border Gateway Protocol version 4) is the de facto standard for inter-domain routing. When considering inter-domain routing, one must consider the interactions between IGP and BGP [29, 30]. Inter-domain MPLS solutions could in theory avoid some of the problems of interaction, but there are still practical complexities in using MPLS in inter-domain routing [26, 27]. Shortest-paths routing cannot be used because it might violate BGP policies. For example, peering agreements typically prohibit transit traffic (i.e. traffic that use backbone B to transit between two points on backbone A), but shortest-path routing allows transit.

There is another problem: traditional traffic engineering algorithms require complete topology and traffic information from all networks. ISPs are typically unwilling to share information such as their topology, link capacities, internal traffic volumes, and routing policies, particularly with potential competitors. As noted in [27] optimization methods which do not have complete information often fall short in performance. Similarly [31] shows that if ISPs cooperate in determining inter-domain routing they can achieve better performance. Can we still attain this improved performance if the ISPs will not share information? It is this problem that we concentrate on here. *How may we perform inter-domain traffic engineering without sharing detailed topological and traffic information?* This is the major difference between our work and the majority of the literature on TE.

The primary problem we consider here is a connected pair of ISPs who wish to optimize the routing of traffic on their joint network. We do not separate the problem into separate intra- and inter-domain TE problems, but regard the joint TE problem. The most closely related works to our own are [31, 32]. Our results agree completely with [31] in that ISPs may gain much larger benefits from TE if they cooperate. We attempt to go further in providing secrecy for the parties. In [31] the providers must reveal opaque preference classes per flow. These certainly hide a great deal of the internal information of a network, but still open the network to indirect inference about its properties if not very carefully implemented. We aim to show just how little information needs to be shared to perform a joint optimization, and the tradeoffs between sharing information and performance.

GATEway is pragmatic in the sense that we aim to solve the problem in a way implementable using current routing protocols without modification. The primary constraint this applies to our work is that we use BGP for inter-domain routing. BGP provides quite good means to control outgoing traffic, but only limited means to control an ISP's incoming traffic. However, if two network operators jointly control their outgoing traffic the effect is control in both directions. In [31] this is achieved through negotiation of the exit points. We shall also aim to control exit points for traffic, though the choices will only be negotiated implicitly. We will refer to the type of routing solution we consider as *pinned-exit routing*, because the ISPs *pin* the exit point of particular flows. However, we will use shortest-path routing within an ISP, and we will not allow path sharing other than across MECPs.

2.2 Privacy Preserving Computation

The problem we consider comes under the heading of *secure distributed computation*, i.e. computing some function of several pieces of data without explicitly combining data (and thus revealing it). Another term used to describe this would be *privacy-preserving multiparty computation* (we use the terms synonymously).

The area of secure distributed computation has been heavily influenced by Yao's two party protocol [33, 34], which is a protocol between two peers that can compute any polynomial-time function pair $(f_x(\mathbf{x}, \mathbf{y}), f_y(\mathbf{x}, \mathbf{y}))$, where $\mathbf{x}$ and $\mathbf{y}$ are the inputs and $f_X(\cdot)$ and $f_Y(\cdot)$ are the functions of interest to the two parties X and Y, respectively. The impressive thing about the protocol is that neither party learns the other's input data, or their output, i.e. X only learns f_X, not f_Y or $\mathbf{y}$. The classic example of Yao's protocol is the computation of the minimum of two values. The protocol requires two rounds of communication and has $O(n)$ computation and communication cost (where the numbers are represented in n bits). However, the protocol is not always efficient, and so many techniques have been developed to improve computational complexity and communications costs for specific problems. This area is now well developed – see [35] for a listing of a number of significant papers. Relatively little work has been done on privacy preserving computation for Internet applications. Brickell and Shmatikov [36] provide an algorithm to solve the shortest-paths through a pair of connected networks, and Machiraju and Katz [32] consider the flow maximization problem for a pair of networks. Note though that these both have polynomial time algorithms for the non-distributed problem. Yao's two party protocol, and related approaches provide methods for computing polynomial time functions. The problems here are NP hard.

Also importantly, note that in some problems, even though an algorithm leaks no side-information, X or Y might still derive information the inputs from the output alone. A good example is the shortest-path problem: the privacy-preserving algorithm for shortest paths on a pair of connected networks is strictly privacy preserving [36]. However, knowledge of the output (shortest-paths) is sufficient to derive information about the weights of the joint network [37]. There is an important distinction between ensuring that the computation is private as opposed to the results being something that the two parties are willing for their partner to know.

On the other hand some of the input data may be easily observable by both parties in any case. For instance, in the shortest-path example when the routing is implemented we could simply measure it. Hence leakage of this information is inconsequential. Given these two features, we do not concern ourselves with strict privacy-preservation here. Instead, we seek to minimize the leakage (by the algorithm or solutions) of information that could not be otherwise observed by the participants. It is no longer a formal, provable definition (as is strict privacy-preservation) but it's consistent with the aims of potential participants in GATEway.

2.3 Assumptions

Most approaches to inter-domain traffic engineering can be characterized as selfish (where one provider acts unilaterally to improve its own performance), or as co-operative where the providers are willing to share information and co-operate (exceptions being [31, 32]). In GATEway we aim to get the best of both worlds. Note, we may still assume that the providers are selfish, but not in quite the same sense meant elsewhere. They will seek to maximize their own gains. However, in the approach we propose, we change the outcome of problems such as the Prisoner's Dilemma by introducing a type of trust. If the prisoners can trust each other, then they can achieve the global optimum. Note that both are still acting selfishly, but given the additional information, the correct selfish choice is also the global optimum.

The model we assume for network operators is sometimes called "semi-honest". It assumes that the providers are not malicious, i.e. they will not deliberately aim to cause damage other network operators, without any positive gain for themselves. They will not act like a "Dog in the Manger" (Aesop). Such participants are sometimes called "honest but curious", because they may seek to find out information, and exploit this information to their own benefit (and possibly to the detriment of other operators). This is a fair assumption because the current Internet relies on this characteristic. One large operator could cause considerable problems for others were they to act maliciously — it has, for instance, happened by accident on more than one occasion.

2.4 GAs

The concept of a Genetic Algorithm (GA) (see [38] and the vast number of publication since) is based on the metaphor of Darwinian evolution — survival of the fittest. The idea, in brief, is to create a population of solutions to a problem, and then let them reproduce and evolve such that we tend to keep better solutions to the problem.

One key advantage of a GA is that the fitness need not be specified in closed form. For instance, GAs are often used in optimizing strategies for games where the fitness is determined by competition between the members of a population. This advantage is key in our application because it allows the parties involved in the computation to share only limited information about fitnesses, rather than the details of each others networks.

We extend the use of biological metaphors in GATEway to the use of the term *symbiosis*. In biology, symbiosis (sometimes mutualism) refers to two different organisms that form a mutually beneficial union. A classical example occurs in coral reefs [39]. Coral polyps are a small colonial organism that build large endo-skeletal reefs out of calcium carbonate. However, they get the majority of their food supply from photo-synthetic algae (called zooxanthellae) which reside inside them, and incidentally provide them with their attractive coloration. The algae gain a safe home, while the coral polyps gain a food supply – both parties benefit from the interaction. Typically such organisms co-evolve to this state, i.e. both evolve together jointly (ancient corals did not exhibit this relationship). Co-evolution is not restricted to symbiotic relationships — it can also occur for competitors for instance — but the point of interest here is that the two organisms don't need to share genetic material to perform such a co-evolution. We may exploit this in GATEway.

3. EVALUATION METHODOLOGY

3.1 Test networks

We have tested GATEway on two sets of topology data. Random networks, and Rocketfuel networks. While we also use random networks to validate GATEway these tests are omitted, primarily because of space restrictions, but also because they are consistent with, and add little to the findings on more realistic topologies.

The Rocketfuel topologies [37] consist of a large number of networks and their peering links mapped primarily using traceroutes. The network maps produced are not perfect, however, they rep-

ASN	Name	PoPs (degree ≥ 2)	links
1	Genuity	24	74
701	UUNet	48	368
1239	Sprint	33	130
2914	Verio	47	176
3356	Level 3	46	536
3561	Cable & Wireless	59	592
7018	AT&T	35	136

Table 1: The Rocketfuel networks used in this study.

resent the best current maps showing both the intra-domain and inter-domain topologies of a significant number of large networks. We concentrate on a group of tier-1 networks, based primarily in North America (though some have significant components in Europe, Asia and the Pacific). We choose these because they all peer with each other with multiple physical connections. In addition, these networks are the largest, and thus provide the best test of the scalability of GATEway. The result is that we consider 7 networks, which each interconnect resulting in 21 possible pairs on which to trial the method. Additionally, there is little point in trying to optimize routing for degree one nodes (there is only one link they can use), and so we eliminate such nodes from the networks under consideration. The networks used, and their parameters are shown in Table 1.

The Rocketfuel data do not contain link bandwidths, and so in the absence of this information, we shall use the simplest possible assumption of equal bandwidth links (as in [27]). One exception to this policy is that we will investigate the impact of varying the peering link capacities because these links are often considerably different from backbone links in a number of respects, as a result of being created through negotiations between multiple parties.

3.2 Traffic generation

The units of traffic we shall manipulate will be *flows*. A flow represents the traffic between some source and destination during some time interval. We shall ignore time dependence here for simplicity. Sources and destinations of traffic in IP networks are groups of IP addresses, often with a common *prefix*. Note though, that the groupings we use here are arbitrarily decided by the network operators, i.e. they do not have to correspond to a particular prefix, customer, router, or other logical structure in the network. The only constraint is that we will not divide flows when routing them, other than across intra-domain MECPs.

For simplicity, we shall use flows aggregated to the level of traffic between PoP pairs. Note that this is not a requirement for the method. In general an operator might wish to conceal the addresses allocated to particular PoPs, or simply the number of PoPs in the network. Hence, they could use arbitrarily de-aggregated prefixes, (for instance break the ISPs address space into /24's), or they could aggregate address space allocated to routers. The choice depends on the balance between complexity and the level of optimization required (finer granularity requires more computation, but perhaps allows a greater degree of optimization).

We need to synthesize traffic matrices for our simulations, and so we extend the simple from [40]. We generate the traffic demand matrix between nodes using a gravity model with randomly chosen local traffic vectors. That is, we generate the independent exponential random variables

$$X^k_{i,m} = \text{ the traffic at PoP } i \text{ in network } m \text{ in direction } k,$$

where $k \in \{\text{in}, \text{out}\}$. The demand matrix elements giving the traffic from i to j in networks m and n are $D^{m,n}(i,j) = X^{(\text{in})}_{i,m} X^{(\text{out})}_{j,n}$. Although this method is extremely simple, it was shown in [40] to match real traffic-matrix statistics well.

3.3 Performance metrics

We evaluate the performance by measuring maximum utilizations. However, the maximum utilization on its own may reveal only the size of the traffic, which is being generated via a randomization process. In order to create a basis for fair comparisons we will output the performance (the maximum utilization) relative to the *measured routing* in the Rocketfuel data. Results are reported as a percentage relative to this maximum utilization (smaller values indicate better performance).

4. WEIGHT OPTIMIZATION USING GENETIC ALGORITHMS

The problem of intra-domain traffic engineering can be expressed thus: find the network routing parameters that balances loads on the existing links in a "beneficial" way. There is a very simple approach to solving the intra-domain traffic engineering problem, namely by using the shortest-path routing with a set of optimized link weights. This has the advantage of being easily implemented using current IGPs.

We call this approach the *shortest-path link-weight optimization problem* and it has been extensively studied [4–12, 15]. Despite the apparent limitation of shortest-paths routing, the method has been shown (for realistic networks) to perform almost as well as the most general approaches to routing available, and to have many other advantages (see Section 2.1 for more details).

Take a network described by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the edges of the graph. We denote the number of nodes in the graph by N and the number of edges by E. We seek to choose a function $w : \mathcal{E} \to I\!R^+$, giving the link weights of each link, such that when we solve the All-Paths Shortest Path (APSP) problem, the solution minimizes the maximum utilization of the links in the network. We use the notation w_e, c_e, and f_e to denote link e's weight, capacity, and load, and the link utilization is defined to be $u_e = f_e/c_e$. Given a set of link weights, the APSP routing is the routing that minimizes for all $i, j \in \mathcal{N}$ the distances $d_{ij} = \sum_{e \in p_{ij}} w_e$ between nodes i and j, where p_{ij} is the set of links along the path chosen between i and j.

The problem of finding an optimal weight setting is NP hard [6], and so we must find heuristic approaches to the solution of the problem. Several proposed heuristic are based on GAs [7,8,11]. We use a slightly different GA here in order to make it easier to generalize to the joint TE problem. The chromosome for each member of the population is a vector containing w_e for each edge. We restrict these elements to be represented by K bits, restricting the range of values to $w_e \in [0, 1, \ldots, 2^K - 1]$. The GA algorithm is then:

1. **initialization:** create (randomly) an initial set of N solutions called the population, $\mathcal{P} = \{\mathbf{x}_i\}$
2. **while** not finished
   a. **evaluate fitness:** $f(\mathbf{x}_i)$ of each $\mathbf{x}_i \in \mathcal{P}$
   b. **generate a new population:** the offspring
      i. **selection:** select two parents from the population based on fitness.
      ii. **crossover:** combine the parents genes to form offspring.
      iii. **mutation:** with a probability q mutate each gene.
   c. **replace old population with offspring.**

However, in designing a GA there is a great deal of flexibility in each of the mechanisms listed here. We take the approach here of using simple techniques with the aim of demonstrating the concept rather than providing the best possible optimization algorithm:

1. **Crossover:** We a single (random) point crossover.

2. **Mutation:** We perform mutation gene by gene independently, with some small probability q.
3. **Selection:** Selection is determined from the fitness function $f(\cdot)$ based on the maximum utilization of a given routing $f(\mathbf{x}_i) = 1/\max_{e \in E} u_e$, and *Roulette Wheel Selection*, i.e., given a set of solutions $\{\mathbf{x}_i\}$, we select a member of the population with probability $p_i = f(\mathbf{x}_i) / \sum_{i \in P} f(\mathbf{x}_i)$.
4. **Termination criteria:** We terminate the algorithm after a fixed number G of generations.

In addition, there are many tweaks one can apply to GAs to improve performance. The only one we use here is *elitism*, i.e. the retention of the best member of the population during each generation with no crossover or mutation. This results in a non-increasing maximum fitness for each generation (a property not guaranteed otherwise).

We use the *measured routing* as an initial value, seeded into the population. This initial value does not have quite the same importance as in many other optimization techniques, because it replaces only one of the initial population. Note we confine our weight values to a smaller range of integers than the Rocketfuel data, so our initial solution may have different routing from the measured routing, and hence our results will not all start at 100% performance.

4.1 Validation of the GA approach

We tested the above approach on a range of simulated networks in order to choose reasonable parameter settings (results omitted because of space restrictions). Our main parameters are the probability of mutation $q = 0.01$, the population size $P = 50$, the number of bits to use in representation of a weight $K = 4$, and 2 elite solutions were retained. We compared our results to those of Fortz and Thorup (FaT) using their code, performing $G = 10000$ iterations for both algorithms. Figure 1 (a) shows the relative performance of our approach, and FaT with respect to the performance on the measured routing on the Rocketfuel networks. Both approaches produce similar improvements (though FaT performs 2% better overall). Figure 1 (b) shows the computation times. The GA times are better by 27% on average. Although these computation times are not insignificant in some cases, weight optimization techniques have a number of advantages. For instance, Roughan *et al.* [15] showed that one could get a large part of the improvement of weight optimization using a much smaller number of iterations, thereby creating a potentially favourable tradeoff between time and performance — we demonstrate the same phenomena in Section 5.3.1. Furthermore, [15] also showed that weight optimization could be performed to create a set of weights that were robust over a period of at least 24 hours (taking into account prediction errors, and daily variations). Hence, significant computation times can be amortised over such periods.

In some cases we observe that the performance of both algorithms was somewhat limited. For instance, in Figure 1, the performance improvement for ASN 7018 was only around 70%. In this particular case we investigated the reason, which was that there were two components of the graph that were poorly connected. In particular, three PoPs in Florida were connected to the rest of the North American nodes via a single pair of links. Given only two links, the opportunities for load balancing are somewhat limited. In the real network this would be reflected in the fact that the two links in questions would either have increased capacity, or the poorly connected network segment would have little traffic. This appears to be a relatively common occurrence in the Rocketfuel topologies, and hence we wished to assess how much our results were biased by such features. To do so, we excise the 3 Florida nodes (and 8 edges) from the AT&T network, and perform the optimization on this new network. The results are shown in Figure 1 under the heading ASN 7018a. Clearly a great improvement was obtained for the reduced network. In the remaining work in this paper we will continue to work with ASN 7018a, the Rocketfuel topology without the Florida nodes, but we leave the other topologies untouched, thus providing some contrast as to the impact of this issue.

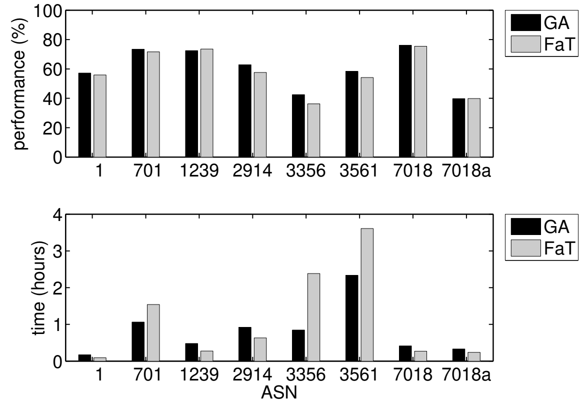

Figure 1: Simple weight optimization using the GA for $G = 10000$, and Fortz and Thorup (FaT) also using 10000 iterations. The results show the mean relative performance for 30 random simulations, and compute times.

4.2 Computational complexity

The algorithm proceeds in a number of iterations G, with population size P, hence its computational cost is proportional to PG, but the critical factor in the computational cost is the cost of evaluating the fitness function, which requires the solution to the APSP problem. We use a simple implementation of the Floyd-Warshall algorithm to perform this step (the algorithm has $O(N^3)$ computational complexity) and Figure 7 confirms cubic complexity.

5. SYMBIOTIC OPTIMIZATION

The previous section considered optimization over only a single network, and similar results have been described elsewhere. We now describe the generalization of this approach to a pair of networks joined together at a set of *peering* links. The GA algorithm is extended to allow joint evolution of two "symbiotic" populations of solutions, one for each ISP. As in biological symbiosis the participants don't have to share all their genetic material. However, there is some information leakage in our initial approach, and we consider how to limit it in Section 6.

5.1 The problem

The problem we wish to solve here is the problem of optimizing the routing of two connected networks. In principle this is no more complex than optimizing one large network (comprised of the two inter-connected networks). However, business constraints restrict the type of routing allowed. For instance, transit routing is not allowed between peers. One peer cannot use another network's backbone to transit its traffic across the country using its own network only at the end points. Hence the simple generalization of shortest-path routing to the joint network created from inter-connecting the two peers will create unacceptable solutions.

Furthermore, as noted earlier, we wish to limit the exchange of information between the two peers. The joint shortest-path solution would require each network to share its topology, and traffic in detail. More precisely, take two networks $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{E}_1)$, and $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{E}_2)$, which are inter-connected by a set of peering links $\mathcal{Q}$, where for $q \in \mathcal{Q}$ we have $q = (n_1, n_2)$ where $n_1 \in \mathcal{N}_1$ and $n_2 \in \mathcal{N}_2$. We can create a joint network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where

$\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$, and $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{Q}$. We shall use the solution to the shortest-path (SP) link-weight optimization problem on a joint network as a basis of comparison, because we have substantial evidence [6, 9, 15] that it will be close to the best possible routing solution. To be clear, in this solution (which we call *joint SP*), the peering links have no special role, and we do not attempt to prevent transit traffic. Hence the solution is an unrealizable idealization, but we use it as a loose lower bound on performance, for comparison.

At the other end of the spectrum, we will also compare results with selfish routing, where each provider optimizes its own routing with information it can measure itself. This *selfish* solution will be poor because each provider cannot anticipate the changes the other will make to its inbound traffic. On the other hand, GATEway

1. can be computed with limited sharing of information;
2. prevents transit; and
3. is reasonably simple to implement with standard routing protocols (e.g. shortest-path IGPs and BGP).

We do this using the mechanism of *exit point pinning*. Given a traffic flow from network 1 to 2, we would choose a particular exit point, and pin this flow so that it uses that exit point. There are a number of mechanism one could use to implement such a pinning (see Section 5.5), and the pinning could be performed at a variety of granularities. As we have previously discussed, we shall consider PoP level flows. We also simplify by pinning based solely on source or destination, not both. In the examples we show source based routing, as it is slightly simpler to explain, though destination based routing (which is an equivalent, though transposed problem) would be easier to implement. For example, traffic from node i in network 1, to node j in network 2, would be pinned to peering link $q(i) \in \mathcal{Q}$ (note we can specify a peering link by its end points $q = (k, m)$, $k, m \in \mathcal{N}$ or its index in the set, e.g. $q = j \in [1, \ldots, Q]$). The exit point chosen for a given traffic flow is not necessarily the closest to the point of origin, so this is not hot-potato routing, but we do not need the full flexibility of a scheme like TIE [17].

Before we can continue, we must also briefly discuss the difference between Origin-Destination (OD) demand matrices, and Ingress-Egress (IE) traffic matrices. As noted earlier we will simulate using an OD demand matrix generated via a gravity model, which specifies the traffic from origin to destination in the joint network $\mathcal{G}$, and so is a $N \times N$ matrix, where $N = N_1 + N_2$ and $N_i = |\mathcal{N}_i|$ is the number of nodes in network i. Denote the OD matrix by D where its elements $D(i, j)$ are the traffic from origin i to destination j, and we can write D in the form

$$D = \begin{pmatrix} D^{1,1} & D^{1,2} \\ D^{2,1} & D^{2,2} \end{pmatrix},$$

where $D^{m,n}$ is the matrix whose elements $D^{m,n}(i, j)$ give the traffic from node i to j in networks m and n.

The IE traffic matrix describes the traffic matrix as seen internally on a single one of the networks, which is not the same as the demands (see [41] for detailed explanations of this phenomena). For instance, for network 1, the observed traffic matrix will not be $D^{1,1}$. Using pinning, we can easily construct an IE traffic matrix $T^{(k)}$ for network k from the OD matrix. We simply take, for example

$$\begin{aligned} T^{(1)}(i,j) &= D^{1,1}(i,j) + \sum_{m=1}^{N_2} D^{1,2}(i,m) I(q(i) = (j,*)) \\ &\quad + \sum_{m=1}^{N_2} D^{2,1}(m,j) I(q(m) = (*,i)), \end{aligned}$$

for all nodes $i, j \in \mathcal{N}_1$, where $*$ is a wildcard, and $I(\cdot)$ denotes an indicator function, i.e. $I(A) = 1$ if A is true, and 0 otherwise. The computation for $T^{(2)}$ is similar. Notice that the matrices $T^{(i)}$ may not follow a gravity model even where D does. Computing $T^{(1)}$ takes $O(N_1^3 + N_1^2 N_2)$ operations, and so the resulting computation is of similar order to the shortest-paths computation. The demands $D^{1,2}$ and $D^{2,1}$ are measurable by either party using flow collection. The internal demands $D^{i,i}$ do not have to be shared.

In addition, we need to be able to compute the traffic on each peering link q, which we can do by

$$\begin{aligned} r_j^{(1,2)} &= \sum_{k=1}^{N_1} \sum_{m=1}^{N_2} D^{1,2}(k,m) I(q(k) = j), \\ r_j^{(2,1)} &= \sum_{k=1}^{N_2} \sum_{m=1}^{N_1} D^{2,1}(k,m) I(q(k) = j), \end{aligned}$$

where $\mathbf{r}^{(1,2)}$ and $\mathbf{r}^{(2,1)}$ are vectors of the loads on peering links. Both providers know the capacity of peering links.

Network operator i can now compute the APSP, and hence compute the internal links loads on network $\mathcal{G}_i$ using only local information: the IE traffic matrices, a set of exit points, and link weights on $\mathcal{E}_i$.

5.2 GA solution

Consider the problem above. We wish to find a solution that limits the sharing of information to the necessary minimum, and yet allows optimization to take place. We shall apply the metaphor of *symbiosis* here, allowing each network to co-evolve without sharing all their genetic material.

We start by specifying the chromosomes — there will be four. For each network we use one chromosome to describe its weights, and another to describe the pinning positions. We separate the two groups of information as we may wish to perform cross-over and mutation in different ways for each type of gene. More specifically, each member of the population will be described by the vectors $\mathbf{w}_i$, and $\mathbf{q}_i$, giving the links weights, and pinned exit points, respectively, for networks $i = 1, 2$. As before, the weights are restricted to $[1, \ldots, 2^K - 1]$ and $q_i \in [1, \ldots, Q]$, where there are Q peering links. Network operator i holds $\mathbf{w}_i$ and $\mathbf{q}_i$. The values of the pinnings are shared, but the network weights are not, thereby keeping secret each networks' internal topology.

Each network uses the traffic matrices, pinnings, and its own internal weights to compute its own internal link utilization, and the peering link utilization. The information necessary to compute the joint fitness function (the maximum utilizations) is shared, so that each network knows the joint fitness of all members of the population. From this each performs selection, sharing the seeds used in pseudo-random number generation such that they each select the same population members. The two then perform cross-over, and mutation independently (only on the chromosomes they hold).

5.3 Evaluation

5.3.1 Performance

We first compare the joint SP optimization, with the symbiotic solution, and selfish pinned-exit routing. We perform 10 realisations of each pair of networks (a total of 210 simulations), and show the Cumulative Distribution Function (CDF) in Figure 2. The performance of the symbiotic approach is very close to that of the joint SP problem (despite that solution being unrealisable). On the other hand the selfish routing results are profoundly worse (in about a third of cases the provider is actually worse off after applying selfish routing). Table 2 summarizes the average performance. Figure 3 shows the performance after each iteration for a specific network pair. Most importantly we learn from this graph that the majority of improvements in performance occur early on in the optimization. Hence, one could find useful tradeoffs between performance and speed. The graphs for other provider pairs also support this view. Additionally, we considered how various characteristics of the networks influenced performance. Figure 4, shows that the performance was correlated with the network size. We speculate that this is because larger networks provide more opportunities for route diversity, which may be beneficial for shortest-path routing optimization (we see a similar phenomena in Section 5.4 for larger networks).

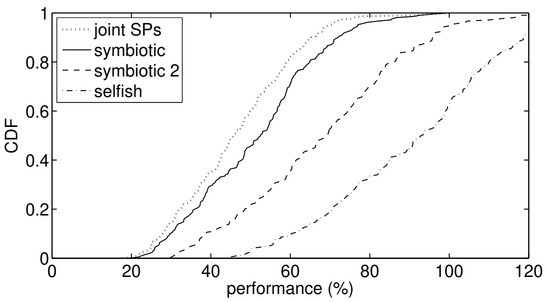

Figure 2: The CDFs showing the performance of the TE techniques with respect to the measured routing with $G = 5000$.

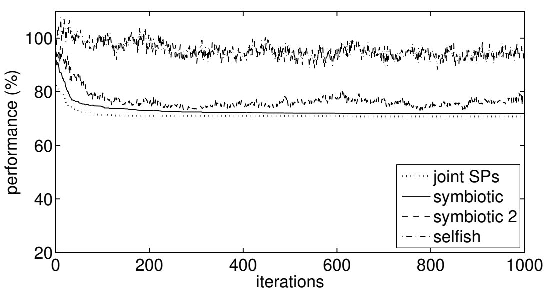

Figure 3: TE across the Rocketfuel AS 1239 and 7018.

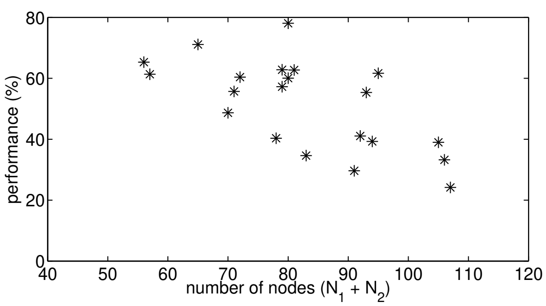

Figure 4: Performance of the symbiotic algorithm with respect to the joint network size.

5.3.2 Peering vs internal links

In the work above, we have deliberately kept things simple in having all link capacities equal. However, anecdotally, peering links are often supposed to be smaller than internal links. Peering links are built through negotiation between competitors. Neither party wishes to pay for the links, and so they are sometimes allowed to reach a state of congestion before any action is taken to upgrade the links. In comparison, anecdotal evidence suggests that most major backbones are relatively lightly utilized, and are likely to remain so under due to the requirements for failover capacity.

Figure 5 shows the relative performance of the algorithm as peering link capacity varies with respect to backbone capacity. The figure shows the maximum link utilization relative to the measured routing for the Rocketfuel networks 1239, and 7018 averaged over 10 simulations. The figure also shows the maximum peering link, and internal link utilizations. For normalized peering link capacities below about 0.4 the performance of the algorithm is dominated by the peering link performance, i.e. the maximum link load occurs on a peering link. Under such circumstances, the relative performance is dominated by a bin-packing problem, which unsurprisingly can be solved significantly better than the measured routing. On the other hand, as the peering capacity increases, the network performance becomes dominated by the internal link capacities. In particular, note that the performance approaches the individual performance of network 1239.

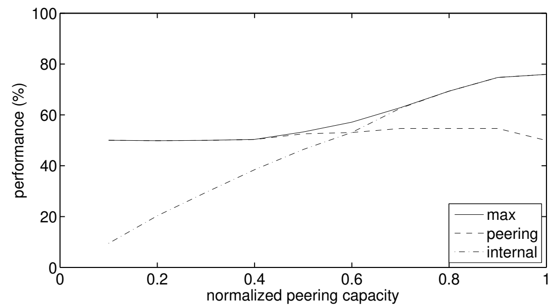

Figure 5: Maximum utilizations of the network, internal links, and peering links as the normalized peering capacity varies.

5.3.3 Alternative metrics

The algorithm above has been shown to find a good min-max link utilization solution to the routing problem. However, network operators may not share this goal; they may wish to optimize other objective functions. A key advantage of GAs is their flexibility with respect to objective functions. We have tested our approach against the metrics drawn from [6, 9] and shown that in this case we also see good performance gains, if anything even faster (results omitted because of space limitations).

5.3.4 Robustness

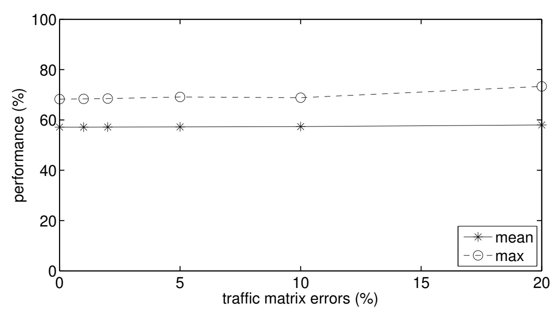

Figure 6: Robustness results for Rocketfuel networks 1239 and 7018, averaged over 100 simulations.

TE is typically applied predictively, i.e. one measures the network, determines the routing to be used, and then this is applied in some future time interval where the traffic may not be identical to that measured. In addition, measurements themselves may contain errors, for instance where sampling or inference is used in data collection. Hence, robustness to measurement or prediction noise is a highly desirable characteristic of any TE algorithm. One of the advantages of optimal weight assignment is robustness to noise [5, 13, 15].

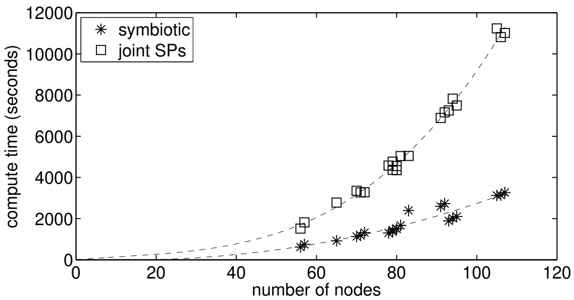

Figure 7: The computation times of the algorithm with respect to number of nodes for the Rocketfuel topologies ($G = 5000$), and fitted cubic polynomials.

We test the robustness of GATEway by determining the optimal (or near optimal) routing using the symbiotic GA, but then measuring its performance on a network where a different traffic matrix is applied. For each initial OD demand matrix $D(i, j)$, we measure performance on a traffic matrix with multiplicative noise, i.e. $D_{\rm err}(i, j) = D(i, j)\,[1 + \sigma N(i, j)]$, where $N(i, j)$ is an independent standard normal random variable, for each i and j, where we vary σ such that the standard deviation of the noise relative to the initial traffic varies from 0 to 20%. For each of the 10 initial traffic matrices we repeat this experiment 10 times, adding different noise each time, for a total of 100 experiments. Figure 6 shows the results for the Rocketfuel networks 1239 and 7018. The figure shows both the average performance, and the worst case performance (max). Even the worst performance over the set of 100 experiments shows great insensitivity to the errors. Similar results are observed for other values of peering capacity. It may seem surprising that the results are quite so insensitive to the input traffic, but this is roughly consistent with the results of [5, 13, 15], which showed remarkable insensitivity to noise in the simple weight assignment problem.

5.3.5 Computational Complexity

The issues surrounding computational complexity of this algorithm are essentially the same as those for the simple intra-domain problem, resulting in $O(N^3)$ complexity. Note though that the size of the network on which we evaluate shortest paths is the individual networks, not the joint network, and so the computational time is $O(N_1^3 + N_2^3)$ which is much faster than the $O((N_1 + N_2)^3)$ computational time for the joint network. Given two equal sized networks the reduction in computation time is a factor of 4. Figure 7 confirms the algorithms' complexities.

5.3.6 Communications Cost

The implementation of this algorithm as a distributed algorithm requires a transfer of information between the two peers. The information to be transferred consists of:

1. The pinning points for each member of the population, for each generation.
2. The information needed to compute the fitness function (in our case, the maximum link utilizations).

The information require to compute fitnesses is small compared to the pinning information. The pinning information requires vectors of size N_i to be transmitted for each network i, for each member of the population, and at each generation. Hence the communication volume is $O(N_{\max} PG)$. Note also that each value to be transmitted is an integer in the range $[1, \ldots, Q]$, where Q is typically small (< 16), and can therefore be represented with around 4 bits without compression. However, after an initial random selection, the pinning vectors are not random, but are the result of a highly non-random process of evolution, and so are quit compressible. We tested this by writing the population of pinning vectors for each network to a file (for the example considered above with the two networks ASN 7018 and 1239), and using gzip to compress the files. Compression ratios of around 4:1 were achieved within 10-15 generations. Thus the pinning data can be communicated with around 2 bits per value. Given parameter values used here (for instance $P = 50$, $Q \sim 10$, $N \sim 50$), the communications cost is < 1 kB per generation.

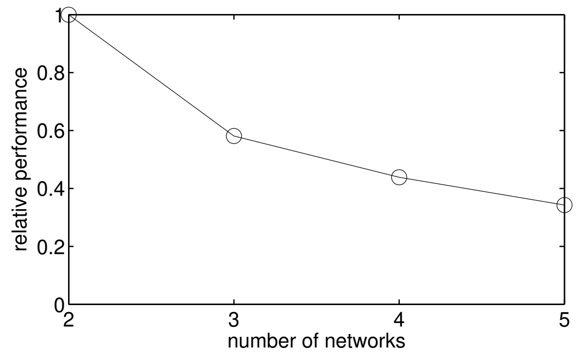

Figure 8: Performance of multi-party GA as a function of the number of participants, relative to performance for two participants.

5.3.7 Other violations of assumptions

The largest assumption in all of this work is the "honest but curious" assumption. It is a fair assumption — the current Internet relies on this as well, given the relatively insecure nature of inter-domain routing at present. However, it is interesting to consider what happens if this assumption is violated. Imagine that one of the ISPs either lies about, or is mistaken in the data it provides to the algorithm, or chooses not to follow the routing determined by the algorithm. It is a simple matter then for the other ISP to measure the traffic across its peering links using flow capture, and from this determine that a problem has occurred. If the problem reduces their performance, then they may either renegotiate a new routing (via our algorithm or otherwise), or go back to their old routing, so they are no worse off than before commencing the use of this algorithm. The other ISP may possibly be better off in the short run through its dishonest behavior, but in the long run they are unlikely to make any more gains than they would by violating current BGP policies.

5.4 Multiple-party optimizations

The extension of this work to more than one party is quite straightforward. N peers (in the sense of neighbouring networks that do not allow transit) can perform the same type of optimization, such that each network retains the information about its own link weights, and shares appropriate pinnings with each peer. Given the GAs ability to cope with arbitrary fitness functions, the generalization is obvious. Figure 8 shows relative performance of the optimization as the number of participants increases. Again this seems to be a result of the increased diversity of routes in a larger network.

5.5 Implementation

The GA would use a protocol independent of the routing protocol. The optimization only requires concrete instantiation in routing once an optimal solution has been determined. There are two approaches to instantiate the derived routing using standard, existing routing mechanisms. Firstly, tunnelling techniques such as MPLS, or IP over IP encapsulation allow explicit choice of exit points. Such techniques have already been proposed for use in [17].

Alternatively, one could use BGP mechanisms to alter exit points.

Mechanisms such as local preference, and MEDs are used to control exit points. These apply control across a whole network (e.g. the exit point for all source nodes for a particular destination would be the same), which implies a destination based pinning. We showed that such a pinning would still provide excellent gains in performance. Even if BGP is used, only exit points are changed, so announcements outside the AS are not needed, and iBGP convergence times will be much shorter than eBGP convergence times.

6. PRIVACY MAXIMIZATION

The above approach to joint network optimization limits information sharing, but there is still some leakage through the pinning vectors and fitness functions. The joint fitness calculation requires the ISPs to share maximum utilization data. This problem is alleviated in part through the use of the utilization metric of [6, 9], but can be improved further.

One of the advantages of the GAs is that the fitness function can be arbitrarily chosen. All we really need to know are the selection probabilities for each member of the population of possible solutions. We have a polynomial-time algorithm for constructing these probabilities, and therefore Yao's two party protocol applies. This is now a well researched area (for instance see some of the reference at [35]), and so, given space limitations, we only briefly describe the approach. There are three steps: firstly, we must solve the APSP for each network, given its internal weights. This can be done internally by each provider. Then these routes must be used to compute the load on each link from the OD demands. For internal links $f_e = \sum_{i,j=1}^{N} D(i,j)I(e \in p_{ij})$. This can be directly computed where $i, j \in \mathcal{N}_k$, but for $i \in \mathcal{N}_k$ and $j \in \mathcal{N}_m$, $k \neq m$ we need to break the indicator into two parts

$$\begin{aligned} I(e \in p_{ij}) &= \sum_k I(e \in p_{ik}) I(q(i) = (k,*)) \\ &\quad + \sum_m I(e \in p_{mj}) I(q(i) = (*,m)), \end{aligned}$$

where $q(i)$ is the peering link for traffic originating at node i. The number of bits for $D(i,j)$ is $O(nN^2)$ where we represent the values with n bits, while for the indicator functions there are $O(EN^2 + N \log Q)$ bits. Yao's protocol's communications cost is linear in the number of bits [34], and so requires $O(nN^2 + EN^2 + N \log Q)$ overhead. The above computation has to be performed for each edge, so given that typically $E > n$, and we can write the complexity as $O(E^2 N^2)$. The third step is to compute the maximum of these values, for which a standard version of Yao's protocol is sufficient, and with comparably negligible overhead (as is the overhead of computing the peering link loads). Additionally, secure operations can be composed, hiding intermediate data. Hence it is possible to perform a step of the symbiotic algorithm which satisfies the definition of privacy preserving, in the sense that the two ISPs need not share (i) utilization data, (ii) pinning data, (iii) any other details of their internal network. We call this solution *privacy-max* and note its performance is the same as the previous symbiotic solution. The cost for using this approach is an increased communications cost associated with performing Yao's protocol.

An alternative to strict privacy preservation via Yao's protocol is to separate selection into the two networks. More precisely, each network computes its own fitness function, and each uses this to select one parent for cross-over. The two networks share the pinning information which is needed to compute link utilizations (again Yao's protocol could be used here to avoid this information being shared). However, the two network use completely independent fitness functions — the fitness functions need not even be the same, thus avoiding any need to share this information. There is a cost in performance. The method (which we refer to as "symbiotic 2"), doesn't perform as well as the simpler algorithm. The results for this independent symbiosis are shown in Figures 2 and 3, and Table 2 summarizes the performance of all methods considered here. The average performance after 5000 generations is 68.4% as compare to 51.5% for the previous algorithm, though still a considerable improvement on the selfish solution. The performance reduction occurs because, although we still use elitism, each network chooses its own elite member of the population without knowledge of the fitness function of the other network. As a result, the chosen elite members of the population are not necessarily elite from the point of view of the other network or a joint fitness function. Hence performance (as measured by the joint maximum utilization) is no longer monotonic. Figure 3 shows this non-monotonicity. The final solution is actually worse than some of the solutions chosen along the way, but without knowledge of the joint fitness, we have no way to know this, and choose the better solution.

Approach	Shared data	Communications cost	Perf.
joint SP	$\mathcal{G}_i$, $\mathbf{c}_i$, $\mathbf{w}_i$, D	$O(N^2 + EK)$	46.6%
symbiotic	$\mathbf{q}_i$, $\max_{e \in E_i} u_e$	$O(GPN_{\max} \log Q)$	51.5%
symbiotic 2	$\mathbf{q}_i$,	$O(GPN_{\max} \log Q)$	68.4%
privacy-max	sel.prob.s $p(\mathbf{x}_i)$	$O(GPE^2 N^2)$	51.5%
selfish	none	zero	91.2%

Table 2: Comparison of approaches. Communications cost for the privacy-maximization approach are a worst case, with the likely cost being significantly smaller.

Note that the results for "symbiotic 2" also illustrate another important point. In these examples we use *different* fitness functions in the two networks. The fitness are computed independently, so this is easily incorporated.

7. CONCLUSIONS AND FUTURE WORK

This paper presents GATEway, a set of algorithms for joint TE between two networks who do not wish to make disclosure of information about their networks. We demonstrate a distinct advantage to combining information, but we present methods here that allow combination of data, without needing to share it. Such approaches could have a significant impact on the way network operators interact.

The approach we have proposed here for a specific problem is actually quite general. It could be applied to other network problems, for instance inter-ISP capacity planning, and perhaps it is also possible to extend these methods outside of the networking world. The important point is that GAs make the approach inherently flexible to a range of problems where information sharing is undesirable.

8. REFERENCES

[1] R. B. Myerson, *Game Theory: Analysis of Conflict*. Harvard University Press, 1997.

[2] D. Mitra and K.G.Ramakrishnan, "A case study of multiservice, multipriority traffic engineering design for data networks," in *Proc. IEEE GLOBECOM*, pp. 1077–1083, 1999.

[3] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS." IETF RFC 2702, September 1999.

[4] F. Lin and J. Wang, "Minimax open shortest path first routing algorithms in networks supporing the SMDS services," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 2, pp. 666–670, 1993.

[5] K. Ramakrishnan and M. Rodrigues, “Optimal routing in shortest-path data networks,” *Lucent Bell Labs Technical Journal*, vol. 6, no. 1, 2001.

[6] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing OSPF weights,” in *Proc. 19th IEEE Conf. on Computer Communications (INFOCOM)*, pp. 519–528, 2000.

[7] M. Ericsson, M. Resende, and P. Pardalos, “A genetic algorithm for the weight setting problem in OSPF routing,” *J. Combinatorial Optimization*, vol. 6, no. 3, pp. 299–333, 2002.

[8] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup, “A memetic algorithm for OSPF routing,” in *Proc. 6th INFORMS Telecom*, pp. 187–188, 2002.

[9] B. Fortz and M. Thorup, “Optimizing OSPF/IS-IS weights in a changing world,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 756–767, 2002.

[10] J. Murphy, R. Harris, and R. Nelson, “Traffic engineering using OSPF weights and splitting ratios.,” in *INTERWORKING*, pp. 277–287, 2002.

[11] L. S. Buriol, M. G. C. Resende, C. Ribeiro, and M. Thorup, “A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing,” *Optimization Online*, 2003.

[12] H. Kaur, T. Ye, S. Kalyanaraman, and K.S.Vastola, “Minimizing packet loss by optimizing OSPF weights using on-line simulation,” in *Proceedings of the 11th International IEEE/ACM Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. MASCOTS-2003*, 2003.

[13] B. Fortz, J. Rexford, and M. Thorup, “Traffic engineering with traditional IP routing protocols,” *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002.

[14] B. Fortz and M. Thorup, “Robust optimization of OSPF/IS-IS weights,” in *Proc. INOC 2003* (W. Ben-Ameur and A. Petrowski, eds.), pp. 225–230, October 2003.

[15] M. Roughan, M. Thorup, and Y. Zhang, “Traffic engineering with estimated traffic matrices,” in *ACM SIGCOMM Internet Measurement Conference (IMC)*, (Miami Beach, FL, USA), pp. 248–258, 2003.

[16] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, New Jersey: Prentice Hall, 1993.

[17] R. Teixeira, T. Griffin, M. G. C. Resende, and J. Rexford, “TIE breaking: Tunable interdomain egress selection,” in *in proceedings of CoNEXT*, October 2005.

[18] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, “IGP Link Weight Assignment for Transient Link Failures,” in *18th International Teletraffic Congress*, (Berlin, Germany), August 2003.

[19] A. Sridharan and R. Guerin, “Making IGP routing robust to link failures.,” in *Networking 2005*, (Waterloo, Canada), May 2005.

[20] H. T. Kaur and S. Kalyanaram, “A connectionless approach to intra- and inter-domain traffic engineering,” in *2nd New York Area Metro Area Networking Workshop*, (Columbia University), September 2002.

[21] N. Feamster, J. Borkenhagen, and J. Rexford, “Guidelines for interdomain traffic engineering,” *ACM SIGCOMM Computer Communications Review*, 2003.

[22] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure, “Interdomain traffic engineering with BGP,” *IEEE Communications Magazine, Internet Technology Series*, May 2003.

[23] B. Quoitin, S. Uhlig, and O. Bonaventure, “Using redistribution communities for interdomain traffic engineering,” in *Quality of Future Internet Services (QoFIS 2002)*, October 2002.

[24] S. Uhlig, O. Bonaventure, and B. Quoitin, “Interdomain traffic engineering with minimal BGP configurations,” in *18th International Teletraffic Congress (ITC)*, September 2003.

[25] C. Pelsser and O. Bonaventure, “RSVP-TE extensions for interdomain LSPs,” October 2002. Work in progress, draft-pelsser-rsvp-te-interdomain-lsp-00.txt.

[26] S. Uhlig and O. Bonaventure, “On the cost of using MPLS for interdomain traffic,” in *Quality of Future Internet Services (QoFIS 2000)*, September 2000.

[27] C. Pelsser, S. Uhlig, and O. Bonaventure, “On the difficulty of establishing interdomain LSPs,” in *Proc. of the 2004 IEEE International Workshop on IP Operations and Management (IPOM)*, (Beijing, China), October 2004.

[28] S. Uhlig, “A multiple-objectives evolutionary perspective to interdomain traffic engineering,” *International Journal of Computational Intelligence and Applications*, vol. 5, pp. 215–230, June 2005.

[29] R. Teixeira, T. Griffin, A. Shaikh, and G. Voelker, “Network sensitivity to hot-potato disruptions,” in *Proceedings of ACM SIGCOMM*, August 2004.

[30] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan, “Traffic matrix reloaded: Impact of routing changes,” in *Workshop on Passive and Active Measurements (PAM)*, 2005.

[31] R. Mahajan, D. Wetherall, and T. Anderson, “Negotiation-based routing between neighboring ISPs,” in *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, (Boston, MA, USA), April 2005.

[32] S. Machiraju and R. H. Katz, “Reconciling cooperation with confidentiality in multi-provider distributed systems,” Tech. Rep. UCB/CSD-04-1345, EECS Department, University of California, Berkeley, 2004.

[33] A. Yao, “How to generate and exchange secrets,” in *Proc. of the 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 162–167, 1986.

[34] B. Pinkas, “Cryptographic techniques for privacy-preserving data mining,” *SIGKDD Explorations*, January 2003.

[35] “Multiparty computations.” `http://www.cs.ut.ee/~lipmaa/crypto/link/mpc/`.

[36] J. Brickell and V. Shmatikov, “Privacy-preserving graph algorithms in the semi-honest model,” in *ASIACRYPT, LNCS*, pp. 236–252, 2005.

[37] N. Spring, R. Mahajan, and T. Anderson, “Quantifying the causes of path inflation,” in *ACM SIGCOMM 2003*, (Karlsruhe, Germany), 2003.

[38] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[39] A. Baker, “Corals adaptive response to climate change shifting to new algal symbionts may safeguard devastated reefs from extinction,” *Nature*, vol. 430:741, p. 741, 2004.

[40] M. Roughan, “Simplifying the synthesis of Internet traffic matrices,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, pp. 93–96, October 2005.

[41] D. Alderson, H. Chang, M. Roughan, S. Uhlig, and W. Willinger, “The many facets of internet topology and traffic,” *Networks and Heterogeneous Media*, vol. 1, pp. 569–600, December 2006.