Yale University
Department of Computer Science

Mechanism Design for Policy Routing

Joan Feigenbaum[1]
Yale University

Rahul Sami[2]
MIT

Scott Shenker[3]
ICSI and U.C. Berkeley

YALEU/DCS/TR-1258
November 2003

This work was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research under Grant N00014-01-1-0795.

[1]Supported in part by ONR and NSF.
[2]Supported by ONR and NSF.
[3]Supported in part by NSF.

Mechanism Design for Policy Routing

Joan Feigenbaum[*]
Yale University
feigenbaum@cs.yale.edu

Rahul Sami[†]
MIT
sami@csail.mit.edu

Scott Shenker[‡]
ICSI and U.C. Berkeley
shenker@icsi.berkeley.edu

Abstract

The Border Gateway Protocol (BGP) for interdomain routing is designed to allow autonomous systems (ASes) to express policy preferences over alternative routes. We model these preferences as arising from an AS's underlying utility for each route and study the problem of finding a set of routes that maximizes the overall welfare (*i.e.*, the sum of all ASes' utilities for their selected routes).

We show that, if the utility functions are unrestricted, this problem is NP-hard even to approximate closely. We then study a natural class of restricted utilities, *next-hop* preferences. We present a strategyproof, polynomial-time computable mechanism for welfare-maximizing routing over this restricted domain. However, we show that, in contrast to earlier work on lowest-cost routing mechanism design, this mechanism appears to be incompatible with BGP and hence difficult to implement in the context of the current internet. Our contributions include a new complexity measure for Internet algorithms, the *dynamic stability*, which may be useful in other problem domains.

1 Introduction

The Internet is composed of many independently-managed subnetworks called domains or *autonomous systems* (ASes). The task of discovering and selecting routes between these ASes is called *interdomain routing*. Currently, the only widely deployed protocol for interdomain routing is the Border Gateway Protocol (BGP); through BGP, a router can learn of routes from neighboring networks, select routes from the multiple alternatives it may learn of, and advertise its selected routes to other networks.

In this scenario, one of the key decisions an AS must make is how to select a route from all the routes it knows of to a particular destination. One frequently studied model has each AS look at some objective metric over the routes, such as the number of ASes the route passes through or the cost of a route, and pick the route which minimizes this metric. In practice, however, ASes want to select a route based on many other criteria, such as commercial relationships or perceived reliability. For example, it is common for an AS to select a route advertised by one of its customers over all other routes. Thus, BGP was explicitly designed to allow ASes to apply their own *routing policies* to the route-selection and route-advertisement processes. This feature of interdomain routing is referred to as *policy-based routing* or *policy routing* for short.

Another aspect of routing that has recently received attention is that of incentives. The participants in the routing process– the ASes, in this case– are independent economic entities, each with its own goals. Thus, they cannot be relied on to follow any specified policy, if they could profit by deviating from that policy. Further, much of the information relevant to selecting good routes, such as costs or connectivity

This work was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research under Grant N00014-01-1-0795.

[*]Supported in part by ONR and NSF.

[†]Supported by ONR and NSF.

[‡]Supported in part by NSF.

information, is known privately to individual ASes; thus, even if there were a central authority capable of enforcing a policy, it could not possibly detect strategic reporting of this information. This paper explores the extent to which one can cope with these strategic issues in a computationally feasible manner.

The *algorithmic mechanism design* approach, introduced by Nisan and Ronen [NR01], was one of the first works to address both incentives and computational complexity. Among other problems, Nisan and Ronen studied a simple routing problem: Given a graph with a distinguished source node s, a distinguished sink node t, and costs associated with each edge, find the lowest-cost path from s to t. The wrinkle in the model is that each edge can strategically lie about its cost. Nisan and Ronen showed how a central authority can compute payments for each edge such that every edge's dominant strategy is to be honest about its cost, yielding a *strategyproof mechanism* for this problem. Later, Hershberger and Suri [HS01] presented a more efficient algorithm to compute the payments required by this mechanism. Archer and Tardos [AT02] studied mechanisms to select a path that minimizes a metric from a broad class, not necessarily the sum of edge costs; this too can be viewed as a variant of lowest-cost routing.

The mechanism-design approach was extended by Feigenbaum *et al.* [FPSS02], who sought lowest-cost routing mechanisms in the context of interdomain routing. Their main contribution was to focus on *distributed* mechanisms, thus adopting the distributed algorithmic mechanism design approach initiated by Feigenbaum, Papadimitriou, and Shenker [FPS01]. Feigenbaum *et al.* [FPSS02] give a strategyproof mechanism for the lowest-cost routing problem, which can be computed by an efficient distributed algorithm. Moreover, they show that this mechanism can be computed by a "BGP-based" algorithm, *i.e.*, an algorithm with similar data structures and communication patterns to BGP, that requires only modest increases in communication and convergence time. Thus, the mechanism is "backward compatible" with BGP, which is critical for any routing algorithm that must be implemented in the current Internet.

All the work on mechanism design for routing has focused on variants of *lowest-cost* routing. In practice, this has two drawbacks: The cost model is oversimplified, and the requirement that all ASes use a lowest-cost routing policy is too restrictive. In this paper, we investigate whether the distributed algorithmic mechanism design approach can be extended to general policy routing. In essence, we look at interdomain routing at a higher level of abstraction: We assume that source ASes have preferences over alternative routes to a destination, but we do not model the *causes* of these preferences. Thus, in our initial model, an AS can express any routing policy, provided that it is based on *some* underlying utility function– it need not arise from the cost of the route but may take into account unspecified, subjective route attributes as well. The goal of the mechanism is to compute routes for every source-destination pair such that the *overall welfare*, *i.e.*, the sum of all ASes' utility for their selected routes, is maximized. The only constraint on the selected routes is that all routes to a given destination must form a tree; this is a very natural constraint in the Internet, where packet forwarding decisions are based only on the destination (not source and destination) of the packet.

Our first result is that, for general preferences, computing an optimal set of routes is NP-hard; it is even NP-hard to compute a solution that approximates the optimum to within a factor of $\mathbf{O}(n^{\frac{1}{4}-\epsilon})$. We prove this result by an approximability-preserving reduction from the Maximum Independent Set problem.

This leads us to consider a restricted class of utility functions, *next-hop preferences*. The restriction is that an AS's utility for a route can only depend on the first hop along that route. This class of utility functions captures preferences arising from customer/provider/peer relationships an AS might have with its neighbors. These commercial relationships are a major motivation for allowing flexible policy routing in BGP, and so this is an interesting class of preferences to study. We show that, for next-hop preferences, the welfare-maximization problem reduces to finding a maximum-weight directed spanning tree to each destination and is hence computable in polynomial time. We derive a strategyproof mechanism for this problem, and show that it can also be computed in polynomial time.

We next ask whether it is possible to implement this mechanism with a distributed, BGP-based algorithm. Unfortunately, we find that this is not the case. In order to prove that a BGP-based implementation is impractical, we refine the model of BGP-based computation given in [FPSS02] and show that any implementation of the welfare-maximizing policy-routing mechanism would be unacceptable, even on Internet-like graphs with small numeric valuations, for two reasons: (1) The selected routes may be long, and hence the routing algorithm may take a long time to converge; and (2) Any change in any AS's utilities may require communication to $\Omega(n)$ nodes, which defeats the rationale of using a path-vector protocol such as BGP. Thus, we conclude that, unlike the lowest-cost routing mechanism of [FPSS02], this mechanism is not easy to implement in the current Internet.

The rest of this paper is structured as follows: We formulate the welfare-maximizing policy-routing problem in Section 2. In Section 3, we prove that, with arbitrary preferences, the problem is NP-hard, even to approximate closely. We then turn to the next-hop preference model in Section 4. We design a strategyproof, polynomial-time computable mechanism, the MDST mechanism, that maximizes welfare in this model. In Section 5, we elaborate the BGP-based computation model and show that the MDST mechanism is hard to implement in this model. Finally, in Section 6, we summarize and present some open questions. Portions of this work appeared in preliminary form in the second author's PhD Thesis [Sam03].

2 The Policy-Routing Problem

The network consists of n Autonomous Systems. For simplicity, we treat each AS as an atomic entity; thus, we model the network as a graph with nodes corresponding to the autonomous systems. The edges in this graph correspond to BGP peering or transit relationships between ASes: We have a directed edge from node a to node b if b advertises its routes to a. In practice, the edges in this graph may vary with the destination under consideration; however, we assume that these edges are identical for routes to any destination.

A *route* from a node i to a node j is simply a directed path, with no cycles, from i to j in the AS graph. The *routing problem* in this network is as follows: For each pair of nodes i and j, we need to select a single route from i to j. Further, we insist that the set of all routes to destination j forms a tree rooted at j. This is a natural restriction when packets are routed one hop at a time (as opposed to being routed in an end-to-end manner, *e.g.*, source-routed). A candidate solution to the routing problem is thus a set of directed trees, one for each destination. The trees for different destinations are independent of each other, and hence it is possible to analyze the model for a single destination.

The basic difference between the lowest-cost routing problem and the policy-routing problem lies in the source of preferences. In the former, the costs incurred by transit carriers result in their preferring routes that do not pass through them; in the latter, ASes have differing preferences over alternative routes, and the constraint that routes form a tree leads to conflicts of interest. There are many reasons why ASes may have real economic preferences for different routes: Two different routes from i to j may lead to differing transit costs, customer satisfaction, or service payments. In this paper, we assume that AS i's preferences among the candidate solutions are dictated entirely by the route from i to j in each solution, independent of the routes from other nodes to j. In a sense, this is complementary to the lowest-cost routing model, in which AS i's utility for a tree depends only on the routes on which it was a *transit* node.

Specifically, we suppose that AS i's preferences for paths can be expressed as a utility function $u_i : \mathcal{P}_{ij} \to \mathbb{R}$, where $\mathcal{P}_{ij}$ is the union of all possible paths from i to j and the *empty path* $\perp$ (which corresponds to solutions in which there is no route from i to j). Only the relative utilities are important, and so we can normalize this function by requiring that $u_i(\perp) = 0$. Further, we assume that, for any route P_{ij} from i to j, $u_i(P_{ij}) \geq 0$; in other words, having any route to j cannot be worse for i than having no route at all.

AS preferences are private information, and hence an AS may misreport its preferences, unless it is given appropriate incentives. These incentives are provided through a *mechanism*. Abstractly, a mechanism for the routing problem for destination j takes as input the user utility profile $\mathbf{u} = (u_1, u_2, \dots, u_n)$ and outputs a routing tree T and a vector of payments $\mathbf{p} = (p_1, \dots, p_n)$, where p_i is the amount of money paid to i. We use the notation $u_i(T)$ to denote i's utility for its path to j in the tree T. We assume that the utility functions are quasilinear and thus can be expressed directly in terms of money. Then, AS i's combined benefit from the mechanism can be expressed as the sum $(u_i(T) + p_i)$. A mechanism is *strategyproof* if the payments are such that every AS i's dominant strategy is to truthfully report u_i. In other words, strategyproofness requires that, regardless of other ASes reported utility functions, each AS i maximizes the sum $(u_i(T) + p_i)$ by reporting its true utility function u_i to the mechanism.

The economic goal of this routing mechanism is to maximize the *overall welfare*, *i.e.*, to choose a routing tree T that maximizes $W(T) = \sum_{i \in P} u_i(T)$. We call this the *welfare-maximizing routing problem*.

3 NP-hardness of the general problem

In this section, we show that the general form of the welfare-maximizing routing problem stated in Section 2 is not tractable. We will focus on computing routes to a single destination only. BGP essentially computes routes to different destinations in parallel, and so it is sufficient to consider the single-destination case. Moreover, our results show that a policy-routing mechanism is hard to compute, even for a single destination. It follows *a fortiori* that it is hard to compute the mechanism when all destinations are considered (although the complexity may not grow by a factor of n).

An instance of the routing problem we are considering is as follows: We are given a directed graph G, with a distinguished destination node j. Each node i is associated with a set S_i of *allowed paths*[1] from i to j in G and a utility function $u_i : S_i \to \mathbb{R}_{\geq 0}$.

We now show that, for the very general class of utility functions defined in Section 2, it is NP-hard to compute a tree that maximizes the overall welfare. We prove this result by a reduction from the *Independent Set* problem: Given a graph G with vertices N, find a largest subset S of N such that no two vertices in S have an edge between them. This problem is known to be NP-hard [Kar72]; in fact, it is even NP-hard to approximate the size of the largest independent set to within a factor of $n^{\frac{1}{2}-\epsilon}$ [Hås99].

Given an instance $G = (N, E)$ of the Independent Set problem, we construct an instance of the welfare-maximizing routing problem. The construction of the network H is illustrated in Figure 1. For each vertex v in N, we have a *terminal vertex* t_v in H. In addition, for each edge $e = (v_1, v_2)$ in E, we add three vertices e^{v_1}, e^{v_2}, and $\overline{e}$ to H. We also add directed edges from $\overline{e}$ to e^{v_1} and e^{v_2}. Finally, we add a special *destination vertex* j to H. We then choose an arbitrary order for the edges in E. For a vertex v in N, let $e_1, e_2, \ldots, e_l$ be the edges incident on v in G, in that order. We add the directed edges $(t_v, \overline{e}_1), (e_1^v, \overline{e}_2), \ldots (e_{l-1}^v, \overline{e}_l), (e_l^v, j)$ to H.

In this manner, we construct a directed path

$$P_v = (t_v, \overline{e}_1), (\overline{e}_1, e_1^v), (e_1^v, \overline{e}_2), \ldots, (\overline{e}_l, e_l^v), (e_l^v, j)$$

for each terminal vertex t_v. Now, we let $S_{t_v} = \{P_v\}$, and $u_{t_v}(P_v) = 1$, for each such vertex. For a nonterminal vertex $\overline{e}$ corresponding to an edge $e = (v_1, v_2)$ in G, we let $S_{\overline{e}} = \{\overline{P}_{v_1}, \overline{P}_{v_2}\}$, where $\overline{P}_{v_1}$ is the suffix of P_{v_1} from $\overline{e}$ to j, and $\overline{P}_{v_2}$ is the suffix of P_{v_2} from $\overline{e}$ to j. We let $u_{\overline{e}}(\overline{P}_{v_1}) = u_{\overline{e}}(\overline{P}_{v_2}) = 0$. Similarly, for a vertex of the form e^v, we let S_{e^v} contain only the suffix of P_v from e^v to j, and let e^v's utility for this path be 0.

[1]There may be an exponentially high number of paths from i to j in the graph (and, indeed, in the Internet). Thus, it might seem that even describing the AS utility functions completely is a hopeless task. However, it is possible that an AS's utility function can be described with polynomial amounts of space. We include a set of allowed paths in the problem description simply to provide one such representation: A path P_{ij} implicitly has utility 0 if it is not in the allowed set. The NP-hardness reduction in this section shows that, even when all ASes have utility functions that can be expressed concisely using this representation, it is NP-hard to find a welfare-maximizing routing tree. Any other concise representation of utility functions with small support would suffice for the reduction described here.

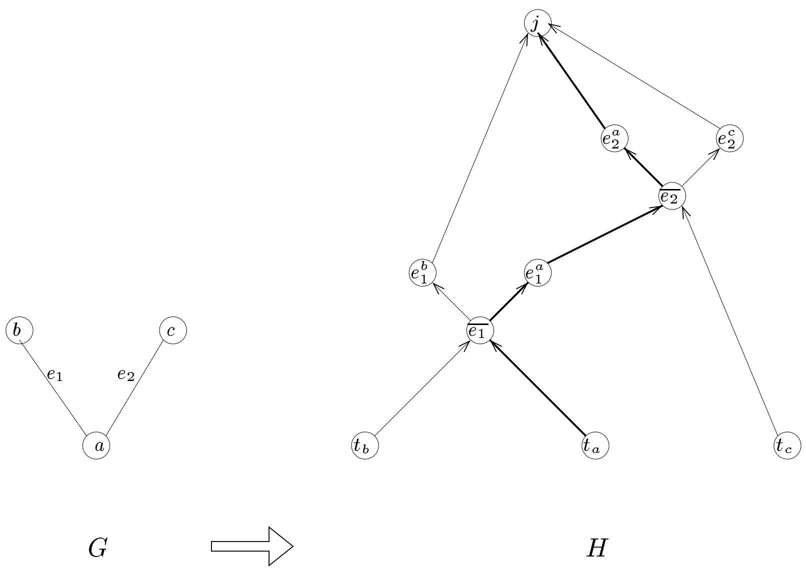


Figure 1: Reduction from Independent Set. The path P_a is shown in bold.

Lemma 1 *Given an instance $G = (N, E)$ of the Independent Set problem, let $(H, \{S_i\}, \{u_i(\cdot)\})$ be an instance of the welfare-maximizing routing problem constructed as described above. Let T^* be an optimal routing tree for this problem. Then, the following conditions hold:*

(i). For any vertices $v_1, v_2 \in N$ such that (v_1, v_2) is an edge in G, at most one of t_{v_1} and t_{v_2} has a path to j in T^*.

(ii). If $S \subseteq N$ is an independent set, then $W(T^*) \geq |S|$.

Proof: (i) Let e be the edge (v_1, v_2). If t_{v_1} has a path to j, it must be the path P_{v_1}. The vertex $\overline{e}$ lies on this path, and hence the unique path from $\overline{e}$ to j in T^* must pass through e^{v_1}, not e^{v_2}. It then follows that the path P_{v_2} is not contained in T^*, and hence there is no path from t_{v_2} to j in T^*.
(ii) No two vertices in S have any edge in common; hence, if $v_1, v_2 \in S$, the paths P_{v_1} and P_{v_2} are disjoint. Thus, the union of paths P_v for all $v \in S$ forms a tree $T(S)$. Further, we note that $W(T(S)) = |S|$. T^* is optimal, and hence $W(T^*) \geq |S|$. $\square$

Corollary 1 *If S is a maximum independent set in G, then $T(S)$ is an optimal routing tree. Conversely, if T^* is an optimal routing tree, then $S = \{v | t_v$ has a path to j in $T^*\}$ is a maximum independent set in G.*

$\square$

Finally, we observe that this reduction implies that even an approximately optimal routing tree is hard to find: If $\tilde{T}$ is an approximately optimal routing tree, then $\tilde{S} = \{v | t_v$ has a path to j in $\tilde{T}\}$ is an approximately maximum independent set in G, with the same approximation factor. Note that we reduce a graph with n vertices to a network with $\mathbf{O}(n^2)$ nodes and $\mathbf{O}(n^2)$ allowed paths. Thus, an $(n^2)^{\frac{1}{4}-\epsilon} = n^{\frac{1}{2}-\frac{\epsilon}{4}}$ approximation to the welfare-maximizing routing problem would give us an $n^{\frac{1}{2}-\frac{\epsilon}{4}}$ approximation to the independent set problem, and an $(n^2)^{\frac{1}{2}-\epsilon} = n^{1-\frac{\epsilon}{2}}$ approximation to the welfare-maximizing routing problem would give us an $n^{1-\frac{\epsilon}{2}}$ approximation to the independent set problem. Combining this with known results on the hardness of computing exactly maximum independent sets and approximately maximum independent sets [Kar72, Hås99], we have proved the following theorem:

Theorem 1 *Given a general network on n nodes, with a total of $\mathbf{O}(n)$ allowed paths and arbitrary AS-path utility functions,*

- *Unless $NP = P$, there is no polynomial-time algorithm to compute a welfare-maximizing routing tree.*
- *For any $\epsilon > 0$, unless $NP = P$, there is no polynomial-time algorithm to compute a tree the total welfare of which approximates that of a welfare-maximizing routing tree to within a factor of $n^{\frac{1}{4}-\epsilon}$.*
- *For any $\epsilon > 0$, unless $NP = ZPP$, there is no polynomial-time algorithm to compute a tree the total welfare of which approximates that of a welfare-maximizing routing tree to within a factor of $n^{\frac{1}{2}-\epsilon}$.*

□

Theorem 1 probably rules out the possibility of solving this problem exactly or approximately in the most general case. There are two possible approaches to restricting the scope of the problem in order to make it more tractable. The first is to restrict the class of networks, while still covering Internet-like situations. The second approach is to restrict the class of allowable utility functions; we pursue the second approach in Section 4.

4 Next-hop preferences

In this section, we consider solutions to the welfare-maximizing routing problem with a restricted class of AS preferences. Specifically, we assume that AS i's utility $u_i(P_{ij})$ for route P_{ij} depends only on the *next hop* from i on this route (*i.e.*, the utility depends only on which of i's neighbors this route passes through). The motivation for this is that an AS is likely to have different economic relationships with different neighbors (customers, providers, and peers), leading to different utilities for routes depending on which neighbor is used for transit; however, it is reasonable to assume that two routes to j through the same neighbor have a similar economic impact on i. Further, we assume that the set of allowed routes from i is likewise determined solely by which neighbors of i may be used to transit packets destined to j.

With this assumption, i's utility function can be written as a function $u_i(a)$ of the neighboring AS a. Similarly, the set of i's allowed routes can be expressed as a set S_i of i's neighbors that can be used to carry transit traffic to j. (The set S_i reflects agreements between i and its neighbors: If $a \in S_i$, it means that, in principle, i is willing to send packets through a, *and* a is willing to accept packets from i for destination j.)

This leads to a convenient combinatorial form of the welfare-maximizing routing problem. We construct a graph G_j, with a vertex corresponding to each AS, and an identified destination vertex j. If $a \in S_i$, we include a directed edge e from i to a; we assign this edge a *weight* $u_e = u_i(a)$. A routing tree is then simply a directed tree (*arborescence*) T with all edges directed towards the root j. Further, an AS i's utility for its route in T is the weight u_e of the edge outgoing from i in T if such an edge exists or 0 otherwise. Thus, the overall welfare with routing tree T is

$$W(T) = \sum_{e \in T} u_e$$

It follows that the welfare-maximizing routing tree T^* is a *maximum-weight directed tree* with root j in G_j.

We first show that we can restrict our attention to directed *spanning trees*.

Lemma 2 *Suppose we are given a weighted graph G_j, with vertex set N. Define $R \subseteq N$ by*

$$R \stackrel{def}{=} \{i \in N \mid \textit{There is a path from } i \textit{ to } j \textit{ in } G_j\} \cup \{j\}$$

Then, there is a maximum-weight directed tree with root j that spans R.

Proof: Let T^* be a maximum-weight directed tree with root j. Suppose there is some vertex $v \in R$ such that $v \notin T^*$. There is a path from v to j in G_j; we can add edges from this path to T^* without decreasing its weight, because the utilities are always non-negative. By adding edges along this path in order, we can eventually grow the tree to include v, without reducing its weight. $\square$

Note that the ASes that cannot even reach j can be completely ignored for the purpose of finding routes to j. Also, it is easy to compute, for each AS i, whether j is reachable from i. This, combined with Lemma 2, means that without loss of generality, we can assume that T^* spans the vertex set N.

Thus, we want to compute a maximum-weight directed spanning tree, with edges directed towards j (a *maximum-weight j-arborescence*).[2] This is a well-studied problem; one distributed algorithm for this problem was given by Humblet [Hum83].

4.1 A VCG Mechanism

We now describe an efficient, strategyproof mechanism for the welfare-maximizing routing problem with next-hop preferences. This is a direct application of the theory of Vickrey-Clarke-Groves (VCG) mechanisms. It follows from the characterization of efficient, strategyproof mechanisms [GL79] that the payment to AS i must have the form:

$$p_i = \sum_{a \neq i} u_a(T^*) \; + \; h_i(\mathbf{u}^{-i}) \tag{1}$$

Here, $h_i(\cdot)$ is an arbitrary function of $\mathbf{u}^{-i}$, the vector of utilities of all agents other than i. We normalize the payment by requiring that nodes that do not carry transit traffic (leaf nodes in T^*) are not paid. The rationale for this requirement here is that leaf nodes are not contributing to other agents' value.

Let T^{-i} be the maximum weight j-arborescence in $N \backslash \{i\}$.[3] Then, $W(T^{-i})$ is a function of $\mathbf{u}^{-i}$ alone. Recall that an AS can refuse to accept transit traffic, *i.e.*, effectively cut off all *incoming* edges. If AS i did this, it would force the optimal tree to have it as a leaf node. We would then have $T^* = T^{-i} \cup (i, a)$, where (i, a), an edge from AS i to some other AS a in the network, is the heaviest outgoing edge from i. As i would be a leaf, the payment p_i must be 0 in this case; for this to occur, we must have $h_i(\mathbf{u}^{-i}) = -W(T^{-i})$. Substituting back into Equation 1, we get the following formula for the payment p_i:

$$p_i \;\; = \;\; \sum_{a \neq i} u_a(T^*) \; - \; W(T^{-i}) = W(T^*) - u_i(T^*) \; - \; W(T^{-i}) \tag{2}$$

We call this the *MDST mechanism.* In order to compute this mechanism, we will have to compute the MDST, as well as the payment p_i to be given to each AS i. The payments can be computed by solving $(n-1)$ minimum-weight j-arborescence instances (one for each node except j), and thus the MDST mechanism is polynomial-time computable.

5 Hardness of BGP-based Implementation

Up to this point, we have formulated the problem of finding the welfare-maximizing routing tree with next-hop preferences as a maximum-weight directed-spanning-tree problem and derived the natural strategyproof, efficient mechanism for this problem. This mechanism is polynomial-time computable in a centralized

[2] This is essentially equivalent to the problem of computing a *minimum*-weight j-arborescence, with weights adjusted appropriately.

[3] We assume the network is 2-connected, and hence such a tree exists.

computational model; this leads us to hope that, as in the case of lowest-cost routing [FPSS02], we can find a BGP-based distributed algorithm for it. Unfortunately, this appears not to be the case. In Section 5.1, we further develop the BGP-based computational model; in sections 5.2 and 5.3, we argue that this mechanism is incompatible with BGP.

5.1 The BGP-based Distributed Computation model

We start by recalling the BGP-based computation model defined by Feigenbaum *et al.* [FPSS02]: An algorithm is "BGP-based" if it has similar data structures and communication pattern to (a simplified abstraction of) BGP. Further, such an algorithm has acceptable performance if the storage space per router, time to convergence, and total communication required in running the algorithm are within a constant factor of the requirements for running BGP itself.

This definition of BGP-based algorithms is not yet complete. For proving that a specific algorithm, such as the price computation algorithm in [FPSS02], does not cause large changes in the structure or performance of BGP, it is adequate– it is possible to assure ourselves by inspection that the algorithm "has similar structure" to BGP. However, for proving impossibility results, we need a more precise specification of the class of acceptable algorithms. Thus, we need to elaborate on the specific properties that we expect a BGP-based computation to have.

Consider routing to some destination j. The properties we require of any BGP-based computation of the routes to j are:

P1 The routing tables should use $\mathbf{O}(l)$ space to store a route of length l.

P2 Routes should be computable in time proportional to the *diameter* of the network rather than the total size of the network.

P3 When a node fails, or there is a change in the information (such as costs or preferences) associated with the node, the change should not always have to propagate to the whole network; instead, it should usually be propagated only to a small subset of nodes. Formally, we require that there are $\mathbf{o}(n)$ nodes that trigger $\Omega(n)$ `UPDATE` messages when one of them fails and comes back up or changes its cost or utility function by an infinitesimal amount.

Property P1 says that the routing table should have roughly the same size as BGP routing tables, or be smaller; this is clearly desirable in any proposed routing algorithm. While the number of ASes in the Internet has grown rapidly, the AS-graph diameter has remained small. In addition, current Internet routes typically pass through few intermediate ASes. Property P2 requires a routing algorithm to stabilize rapidly in networks of this form.

The justification for Property P3 is as follows: In a *link-state* routing protocol, any change has to be broadcast to all the nodes in the network. BGP is a *path-vector* protocol, partly to avoid this dynamic communication burden; thus, a BGP-based algorithm should preserve this property. As the set of routes to j forms a tree, we cannot prevent changes in a few nodes near the root from affecting many other nodes. Similarly, it seems acceptable that a large change in the cost or preference of node i can put it near the root and hence affect many nodes. However, we don't want *every* change to result in this much communication; this is expressed in the statement of P3.

Property P3 is an unusual feature of our model in that it deals with the dynamic performance of an algorithm– specifically, it requires the algorithm to have *dynamic stability*. The main analytic reason for introducing this constraint is to rule out algorithms that compute routes in a centralized fashion at a single location, using logarithmic-depth spanning trees to collect the inputs and distribute the outputs. Such an algorithm is clearly not similar to BGP, yet it could meet the static performance requirements with some clever encoding in the routing tables. The dynamic stability requirement prevents this, and also provides new insight as to why a fully distributed algorithm, such as BGP, may be preferable in loosely-coupled systems.

It may be argued that requirements P2 and P3 capture desirable properties of distributed algorithms generally and not BGP-based algorithms in particular. This is not an obstacle for our purposes in this section. Because we are trying to show that the MDST mechanism is *not* BGP-compatible, it suffices to show that it does not have properties required for a larger class of algorithms that contain those that are BGP-based. These three properties that suffice for the negative result sought in this section. We do not claim that these properties provide us with a fully fleshed out "BGP computational model"; that is a goal for future work.

Another important point is that we do not necessarily require these conditions to hold for all possible networks and all possible cost or preference values. The only networks that we care about are "Internet-like" networks– those that can plausibly represent an AS graph or some subgraph of an AS graph. For this reason, we restrict ourselves to networks that satisfy three properties: They must be sparse, with average node degree $\mathbf{O}(1)$; they must have small diameter– specifically, diameter $\mathbf{O}(\log n)$; and, when any one node is removed from the network, the diameter must remain $\mathbf{O}(\log n)$.

It is more difficult to identify what "reasonable" cost or preference values might be. We definitely want them to be polynomial in n and preferably polylogarithmic in n. Further, we are not as concerned with hardness that may arise because of some strange coincidence of specific numerical values that happen to produce a very unstable state. At the same time, there is no single natural distribution with respect to which we can analyze the average-case complexity of an algorithm. Instead, we insist that any hardness result hold over an open set of cost or preference values; this means that the hardness holds over a region of preference space with non-zero volume, instead of at isolated points. This is similar in spirit to the *smoothed analysis* of Spielman and Teng [ST01].

We note that the lowest-cost routing mechanism in [FPSS02] satisfies these properties, provided the node costs are similar to each other, not very skewed. By contrast, we now show that the welfare-maximizing routing problem does not satisfy these properties, even for networks and preference values that fit our definition of "reasonable."

5.2 Long convergence time

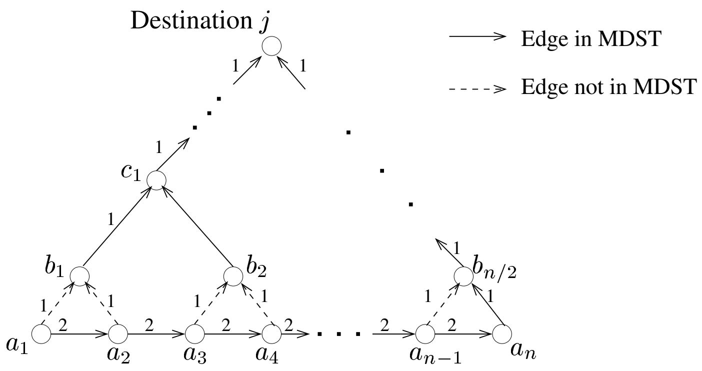


Figure 2: Network with low diameter and a long path in MDST.

Figure 2 shows an example of a network with $2n - 1$ nodes for which a BGP-based algorithm for the welfare-maximizing routing mechanism takes $\Omega(n)$ stages to converge. The network consists of a balanced

j-arborescence. The leaf nodes are $a_1, a_2, \ldots, a_n$. The network can be extended to have diameter $2 \log n$ by adding reverse edges with lower preference values; these reverse edges do not affect our argument, and so we omitted them from Figure 2. Similarly, by adding one more low-preference edge from each internal node, we can arrange for the diameter to remain small even when any one node is removed. Each node is adjacent to at most 4 other nodes, and so the network satisfies the sparseness requirement as well.

The preference values are shown as numbers (weights) on the edges in Figure 2. Each a_i in $\{a_1, a_2, \ldots, a_{n-1}\}$ prefers to route through its neighbor a_{i+1} (value 2) rather than take the path up the tree (value 1). Thus, the welfare-maximizing routing solution, given by the maximum-weight directed spanning tree in this network, consists of the path $a_1 a_2 \cdots a_n$, attached to the remainder of the tree at a_n. Note that the values are in a small range $[1, 2]$. We also remark that this remains the optimal solution even if any subset of the next-hop values are perturbed by a small amount (less than 0.5 each).

Thus, the optimal solution has a route of length $\Omega(n)$, for any preference values in an open set around the specified values. BGP builds routes on a hop-by-hop basis. An AS can use a route only when its next hop on the route has advertised it, and it can itself extend and advertise the route only in the next stage. Thus, we have proved that any such algorithm does not satisfy property P2:

Theorem 2 *Any BGP-based algorithm for computing the next-hop welfare-maximizing mechanism in the network of* $Figure\ 2$*, over an open set of preference values in a small range, takes* $\Omega(n)$ *stages to converge.*

□

Given the hop-by-hop route construction in BGP, it may seem that a more reasonable requirement than P2 is that the number of stages required for convergence is proportional to the length of the longest route. However, the length of the longest selected route is also a function of the mechanism under consideration (in this case, the MDST mechanism); for this reason, we prefer the more stringent requirement P2, which is independent of the mechanism. One of the reasons that the MDST mechanism is incompatible with BGP is precisely that it may select very long routes even in networks with small diameter and hence will cause BGP (or any hop-by-hop protocol substrate) to converge very slowly.

5.3 Extensive dynamic communication

It may be argued that the long route in Figure 2 is unlikely to arise, because long routes are inherently undesirable, and hence ASes will lower their preference values for neighbors with long routes to the destination. In other words, even though next-hop preferences may adequately capture an AS's preferences at any given time, these preferences will themselves evolve (over a longer time period, perhaps) to rule out value profiles that lead to long routes. In this section, we show that, even if there are no long routes, any algorithm to compute the next-hop welfare-maximizing mechanism will not satisfy condition P3: There are situations in which every change in a single node's utility function will trigger update messages to at least half of the other nodes.

We show this by constructing a network as depicted in Figure 3. The network has $n = 2^m + 1$ nodes. We construct it with by recursively constructing clusters of nodes.

At the bottom, we construct a 1-cluster that consists of two nodes, B and R. The 1-cluster has two edges, a "blue" edge from R to B and a "red" edge from B to R. Here, "blue" and "red" are simply labels that we attach to the edges to clarify the analysis; they have no particular semantics. Each of these two edges has weight $L - 1$, where $L = 2m + 4$.

In each cluster in our construction, we identify two special nodes: One is the "blue port" and one is the "red port." For a 1-cluster, B is the blue port and R is the red port. We recursively construct $(k+1)$-clusters from two k-clusters, for $k = 1, 2, \ldots, m-1$: We add a blue edge from the blue port of the right k-cluster to the blue port of the left k-cluster; the latter then serves as the blue port of the $(k+1)$-cluster. Similarly,

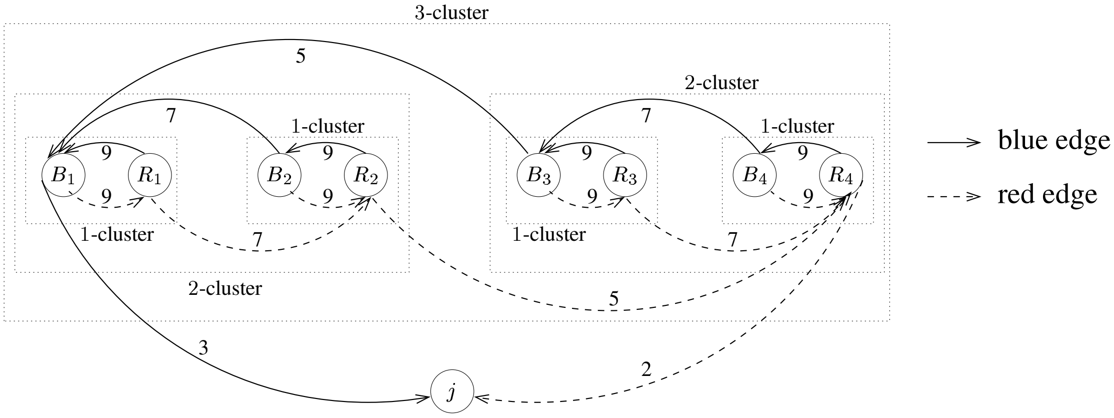


Figure 3: Construction of network for Section 5.3, for $m = 3$.

we add a red edge from the red port of the left k-cluster to the red port of the right k-cluster, which serves as the red port of the $(k+1)$-cluster. These edges both have weight $L - 2k - 1$.

Once we have built up the m-cluster in this manner, we complete the network construction as follows: We add one more node, the destination j. We also add a blue edge from the blue port of the m-cluster to j, with weight $L - 2m - 1 = 3$, and a red edge from the red port of the m-cluster to j, with weight $L - 2m - 2 = 2$. The complete network, for $m = 3$, is shown in Figure 3.

This network is sparse (each node has only two outgoing edges) and has low diameter, as required. As in Section 5.2, we can augment it with edges of lower value so that the diameter stays low on removing an edge; these edges do not affect the analysis, and so we ignore them here. All the valuations are in the range $[1, L]$, where $L = \mathbf{O}(\log n)$. The network we have just built has two distinguished directed spanning trees to destination j: one consisting of all the blue edges and one consisting of all the red edges. In each of these trees, the longest path (route) has $m + 1 = \mathbf{O}(\log n)$ hops. We will now show that these two directed spanning trees have greater weight than any other directed spanning tree with destination j.

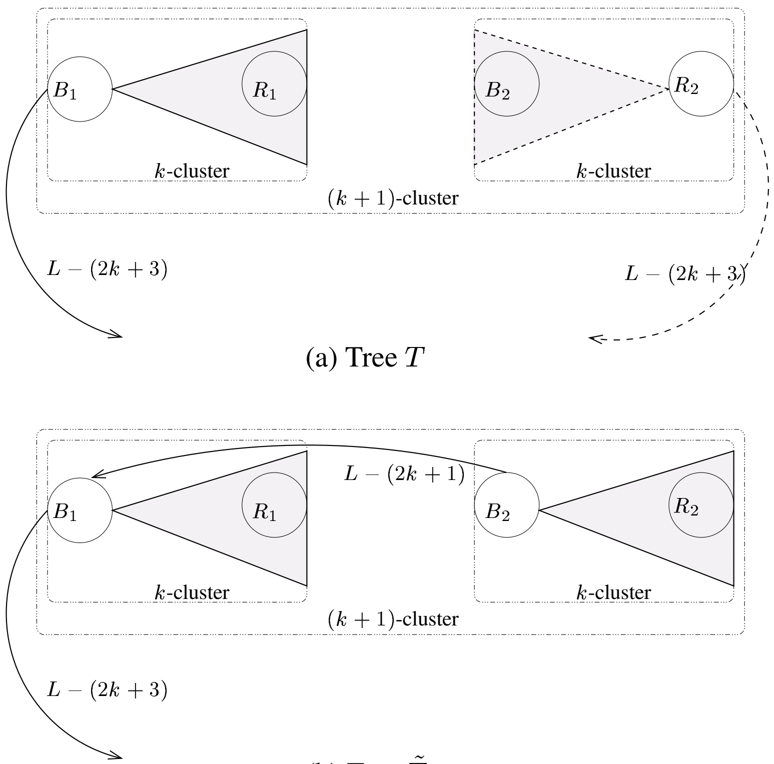
Lemma 3 *If T is a j-arborescence in a network of the form shown in Figure 3, and T has both blue and red edges, then there is another j-arborescence $\tilde{T}$ such that $W(\tilde{T}) \geq W(T) + 2$.*

Proof: Consider a minimum-sized cluster that has both red and blue outgoing edges in T. Suppose this is a $(k+1) - cluster$, as shown in Figure 4(a). Consider the two k-clusters it is composed of, and label the ports B_1, R_1, B_2, R_2 as shown.

Now, the $(k+1)$-cluster has a blue outgoing edge; it must be from the blue port B_1. All smaller clusters have only one color of outgoing edge in T. It follows that the left k-cluster must have only blue edges. Similarly, the red outgoing edge must be from the port R_2, and so the right k-cluster must have all red edges. Thus, the spanning tree T must include the blue spanning tree of the left k-cluster, the red spanning tree of the right k-cluster, and the two outgoing edges with weight $L - 2k - 3$ (or less if $k = m - 1$).

We now construct the tree $\tilde{T}$ as shown in Figure 4(b): we replace the red spanning tree by a blue spanning tree, and replace the red outgoing edge by the blue edge within the $(k+1)$-cluster, with weight $L - 2k - 1$. Because of the symmetric construction of the k-clusters, the red and blue spanning trees have the same weight. Thus, the overall weight of $\tilde{T}$ is at least 2 higher than the weight of T. □

Lemma 4 *For the network and weights $\mathbf{u}$ as constructed in Figure 3, the maximum-weight j-arborescence $T^*(\mathbf{u})$ is the blue spanning tree. Further, for any node B_x that is the blue node of its 1-cluster, $T^{-B_x}(\mathbf{u})$ (the maximum-weight j-arborescence on $N \backslash \{B_x\}$) is the red spanning tree restricted to $N \backslash \{B_x\}$.*



(a) Tree T

(b) Tree $\tilde{T}$

Figure 4: Construction to increase the weight of a tree T with both red and blue edges.

Proof: From Lemma 3, we know that the maximum weight j-arborescence must be either entirely blue or entirely red. At the top level, the blue edge has a higher weight than the red edge; at all other levels of the construction, the weights are the same. Thus, the blue spanning tree must be the maximum-weight j-arborescence $T^*(\mathbf{u})$.

The red spanning tree has B_x as a leaf and has weight only 1 less than optimal. Any other j-arborescence with B_x as a leaf must have both red and blue edges and hence have weight at least 2 less than optimal, by Lemma 3. Finally, we observe that any j-arborescence on $N\backslash\{B_x\}$ can be extended to a j-arborescence that has B_x as a leaf, by adding the red edge (B_x, R_x) with weight $L-1$. Thus, the restriction of the red subtree to $N\backslash\{B_x\}$ must be optimal. □

Now, consider perturbing the weights $\mathbf{u}$ by adding an amount δ_e to the weight of each edge e, for any δ_e with absolute value less than $\frac{1}{n}$. Then, the weight of any spanning tree cannot change by 1 or more, and so Lemma 4 still holds. This leads us to the hardness result for this section:

Theorem 3 *For the network constructed in Figure 3 and an open set of valuations in a small range, any infinitesimal change in valuation must cause* `UPDATE` *messages to be sent to at least* $(n-3)/2$ *nodes.*

Proof: We start with the weight vector $\mathbf{u}$. A perturbed weight vector $\tilde{\mathbf{u}}$ can be constructed from $\mathbf{u}$ as follows: For each node i, we add δ_i^{blue} to the weight of the blue outgoing edge from i and δ_i^{red} to the weight of the red outgoing edge from i, where $|\delta_i^{\text{blue}}|, |\delta_i^{\text{red}}| < \frac{1}{n}$. This corresponds to picking a weight vector from an open set around $\mathbf{u}$.

Consider the payment p_{B_x} due to some node B_x. Let k be such that B_x is the blue port of a k-cluster, but not the blue port of a $(k+1)$-cluster. Then, the blue outgoing edge from B_x has weight $(L-2k-1)$.

The red outgoing edge from B_x must have weight $(L-1)$, and so using Lemma 4 and Equation 2, we get

$$\begin{aligned} p_{B_x} &= W(T^*) - u_{B_x}(T^*) - W(T^{-B_x}) \\ &= W(\text{blue spanning tree}) - (L-2k-1) - [W(\text{red spanning tree}) - (L-1)] \\ &= \Big[W(\text{blue spanning tree}) - W(\text{red spanning tree})\Big] + 2k \\ &= \Big[1 + \sum_{i \in N} (\delta_i^{\text{blue}} - \delta_i^{\text{red}})\Big] + 2k \end{aligned} \tag{3}$$

Note that p_{B_x} satisfies Equation (3) for any perturbed weight vector $\tilde{\mathbf{u}}$ in the given range. Now, suppose we start from some weight vector $\tilde{\mathbf{u}}$, and then there is an infinitesimal change in δ_a^{blue} (or δ_a^{red}) for some node a. It follows from Equation (3) that p_{B_x} changes when this happens, and hence node B_x must receive an update message (or else, it cannot update its value of p_{B_x}). This is true for every blue node, and thus an infinitesimal change in any node's preference must cause price updates at every blue node (a total of $\frac{n-1}{2}$ nodes). Apart from the node a that originated the change (which may be a blue node), every other blue node must receive an update message, thus proving the theorem statement.[4] □

Theorem 3 shows the essence of why the MDST mechanism appears difficult for a BGP-based computational model: A small change at any one node can cause changes that are global, not confined to the routes the node lies on. This appears to be an inherent problem of the maximum-weight directed spanning tree structure: Even if we neglected the payment computation, the failure of any blue node would force the red spanning tree to be used, effectively changing the routes of all other nodes.

6 Conclusion

In summary, we presented a formulation of welfare-maximizing policy routing in the mechanism design framework. We showed that in the most general case, it is NP-hard to maximize the overall welfare, or even approximate it to within any reasonable factor. When utility functions are restricted to the class of next-hop preferences, an optimal strategyproof mechanism is polynomial-time computable. However, a BGP-based distributed implementation of this mechanism appears to be unrealistic: It may converge very slowly even on small diameter networks, and it may require messages to be sent to a large fraction of the nodes whenever any node changes its preferences.

This raises several natural questions for further study. We can ask whether it is possible to design a mechanism for the next-hop preference setting that *approximately* maximizes the overall welfare and also has a low-complexity BGP-based distributed implementation. Another approach is to find reasonable additional restrictions on the preferences for which an efficient exact algorithm exists.

An unusual feature of our computational model is the use of the dynamic communication requirement as a complexity measure. This is possibly relevant to other problem domains as well: Many network protocols are designed to operate over long periods of time, during which their inputs frequently change. Thus, it may be useful to extend the dynamic-stability analysis in Section 5.3 to other distributed optimization problems.

[4] We assume here that the payment p_{B_x} must be stored at B_x. Even if this is not true, we could get a result that is nearly as strong, as follows: p_{B_x} must be stored at *some* node. By property P1, each node can store $\mathbf{O}(m)$ values only; thus, the payments for all the blue nodes must be distributed across $\Omega(n/m) = \Omega(\frac{n}{\log n})$ nodes, which must all receive `UPDATE`s every time the preferences change.

References

[AT02] Aaron Archer and Éva Tardos. Frugal path mechanisms. In *Proceedings of 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 991–999. ACM Press/SIAM, New York, January 2002.

[FPS01] Joan Feigenbaum, Christos Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001.

[FPSS02] Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, and Scott Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02)*, pages 173–182. ACM Press, New York, 2002.

[GL79] J. Green and J. Laffont. Incentives in public decision making. In *Studies in Public Economics*, volume 1, pages 65–78. North Holland, Amsterdam, 1979.

[Hås99] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[HS01] John Hershberger and Subhash Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings of the 42nd annual ACM Symposium on the Foundations of Computer Science (FOCS '01)*, pages 129–140. ACM Press, New York, 2001.

[Hum83] Pierre A. Humblet. A distributed algorithm for minimum weight directed spanning trees. *IEEE Transactions on Communications*, COM-31(6):756–762, June 1983.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations (Proceedings of a Symposium on the Complexity of Computer Computations)*, pages 85–103. Plenum Press, New York, 1972.

[NR01] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[Sam03] Rahul Sami. *Distributed Algorithmic Mechanism Design*. PhD thesis, Yale University, 2003.

[ST01] Daniel Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC '01)*, pages 296–305. ACM Press, New York, July 2001.