Internet routing between autonomous systems: fast algorithms for path trading[☆]

André Berger[a,*], Heiko Röglin[b], Ruben van der Zwaan[a]

[a] *Department of Quantitative Economics, Maastricht University, P.O. Box 616, NL–6200 MD Maastricht, The Netherlands*
[b] *Department of Computer Science, University of Bonn, Germany*

Abstract

Routing traffic on the internet efficiently has become an important research topic over the past decade. In this article we consider a generalization of the shortest path problem, the *path–trading problem*, which has applications in inter–domain traffic routing. When traffic is forwarded between autonomous systems (ASes), such as competing internet providers, each AS selfishly routes the traffic inside its own network. Efficient solutions to the path trading problem can lead to higher global performance in such systems, while maintaining the objectives and costs of the individual ASes. First, we extend a previous hardness result for the path trading problem. Moreover, we provide an algorithm that finds all Pareto-optimal path trades for a pair of two ASes. While in principal the number of Pareto-optimal path trades can be exponential, in our experiments this number was typically small. We use the framework of smoothed analysis to give a theoretical explanation for that fact. The computational results show that our algorithm yields far superior running times and can solve considerably larger instances than a previously known algorithm.

Keywords: Shortest Paths, Internet Routing, Smoothed Analysis

[☆] An extended abstract of this paper has appeared in the conference proceedings of SEA 2011 [3].

[*] Corresponding author: Phone: +31 433884894, Fax: +31 433882000

Email addresses: a.berger@maastrichtuniversity.nl (André Berger), roeglin@cs.uni-bonn.de (Heiko Röglin), r.vanderzwaan@maastrichtuniversity.nl (Ruben van der Zwaan)

1. Introduction

The Border Gateway Protocol (BGP) serves as the main routing protocol on the top level of the Internet and ensures network reachability among autonomous systems (ASes). When traffic is forwarded from a source to a destination, these ASes cooperate in order to provide the necessary infrastructure needed to ensure the desired services. However, ASes also compete and therefore follow their individual strategies and policies when it comes to routing the traffic within their own network. Such locally preferable routing decisions can be globally disadvantageous. Particularly, the way how one AS forwards traffic and through which node another AS may therefore receive the traffic can make a huge difference in the costs for that other AS. Behaving selfishly usually means that an AS routes its traffic according to the least expensive route, also known as hot-potato routing, without regarding the costs of the next AS in the BGP path. That ASes demonstrate such behaviour is supported by the results of Teixeira et al. [18].

Quite a number of protocols have been suggested that require the exchange of information and coordination in order to overcome global suboptimality, while at the same time improving the costs for each individual AS [8, 9, 19]. Recently, Shavitt and Singer [14] considered the case where ASes might be willing to *trade* traffic in such a way that the costs for both ASes do not increase w.r.t. the hot-potato routing, and term this problem *path trading*. They prove that the problem of deciding whether there is a feasible path trade is weakly NP-hard when two ASes are considered. Moreover, they show that there is no constant-factor approximation algorithm for the path trading problem unless $P = NP$. Further, they develop an algorithm based on dynamic programming to find the "best" trading between a pair. Lastly, they give experimental evidence that path trading can have benefits to autonomous systems.

In this article we extend their work in several ways. First, we show that path trading is also strongly NP-hard when an arbitrary number of ASes is considered instead of just two ASes. This justifies the approach taken by Shavitt and Singer as well as the approach taken in this paper to concentrate on path trades between pairs of ASes. We then propose a new algorithm for finding path trades between pairs of ASes that is based on the concept of *Pareto efficiency*. We have implemented both, our algorithm and the algorithm of Shavitt and Singer, and tested them on real Internet instances stemming from [13]. Besides the added advantage that our algorithm obtains *all* Pareto-optimal path trades, it is very fast and

has low memory consumption. As the problem is NP-hard, we cannot expect that the algorithm performs well on all possible inputs. However, in order to support the experimental results we consider our algorithm in the framework of *smoothed analysis*, which was introduced in 2001 by Spielman and Teng [17] to explain why many heuristics with a bad worst-case performance work well on real-world data sets. We show that even though there are (artificial) worst-case instances on which the heuristic performs poorly, it has a polynomial expected running time on instances that are subject to small random perturbations. After its introduction, smoothed analysis has been applied in many different contexts (see [16] for a nice survey).

Finding path trades can be viewed as an optimization problem with multiple objectives that correspond to the costs of the different ASes. A *feasible path trade* is then a solution that is in every objective at least as good as the hot-potato routing. We say that such a path trade *dominates* the hot-potato routing if it is strictly better in at least one objective. This brings us to the well-known concept of *Pareto efficiency* or *Pareto optimality* in multiobjective optimization: A solution is called *Pareto-optimal* if it is not dominated by any other solution, that is, a solution is Pareto-optimal if there does not exist another solution that is at least as good in all criteria and strictly better in at least one criterion. We call the set of Pareto-optimal solutions *Pareto set* or *Pareto curve* for short.

Then the question of whether there is a feasible path trade can simply be formulated as the question whether the hot-potato routing is Pareto-optimal or not. This immediately suggests the following algorithm to find a feasible path trade: Enumerate the set of Pareto-optimal solutions, and then either output that there is no path trade if the hot-potato routing belongs to the Pareto set, or output a Pareto-optimal solution that dominates the hot-potato routing if it is not Pareto-optimal. Also, finding the Pareto set gives the flexibility to choose a solution based on preference. While some solutions might offer great global gain, these trade-offs might be unreasonable from a fairness perspective.

The aforementioned algorithm only works when the Pareto set is small because otherwise the computation becomes too time consuming. Our experiments show that the number of Pareto-optimal path trades is indeed small and that despite the NP-hardness result by Shavitt and Singer we can solve this problem efficiently in practice for two ASes.

For path trading between an arbitrary number of ASes, however, there is little hope for obtaining such a result: We show that our strong NP-

hardness result implies that this problem cannot be solved efficiently even in the framework of smoothed analysis.

Related Work. The potential benefits of collaboration between neighboring ASes and the necessary engineering framework were first introduced by Winick et al. [19]. They consider the amount of information that needs to be shared between the ASes in order to perform mutually desirable path trades and how to limit the effect of path trades between neighboring ASes on the global flow of traffic. The first heuristics for path trading to improve the hot-potato routing were evaluated by Majahan et al. [9]. Majahan et al. also developed a routing protocol that provides evidence that path trading can improve global efficiency in Internet routing. Other related work in the area of improving the global performance while maintaining the objectives of the different ASes has been done by Yang et al. [20], Liu and Reddy [8], and by Quoitin and Bonaventure [11]. Since ASes usually compete, one cannot expect them to reveal their complete network and cost structure when it comes to coordinating the traffic between the ASes. This aspect is considered in the work by Shrimali et al. [15], using aspects from cooperative game theory and the idea of Nash bargaining. Goldenberg et al. [6] develop routing algorithms in a similar context to optimize global cost and performance in a multihomed user setting, which extends previous work in that area [1, 5, 12].

2. Model and Notation

The model is as follows. We have the Internet graph $G = (V, E)$, where every vertex represents a point/IP address. Further, there are k ASes and the vertex set V is partitioned into mutually disjoint sets $V_1, \ldots, V_k$, where V_i represents all points in AS i. We denote by E_i all edges within AS i, that is, the set of edges E is partitioned into $E_1, \ldots, E_k$, and the set of edges between different ASes. The graph G is undirected, and each edge $e \in E$ has a length $\ell(e) \in \mathbb{R}_{\geq 0}$. The traffic is modeled by a set of *requests* R, where each request is a triple (s, t, c), where $s \in V$ and $t \in V$ are source and sink nodes, respectively, and $c \in \mathbb{R}_{\geq 0}$ is the cost of the corresponding request. The BGP protocol associates with each request a sequence of ASes which specifies the order in which the request has to be routed through the ASes. Since most of the paper is about the situation between *two* ASes, we leave this order implicit. The cost of routing a request with cost c through edge e is $\ell(e) \cdot c$. For simplicity, the costs of routing a packet between two ASes are

assumed to be zero, but each request can be routed at most once from an AS to the next AS. The input for PATH TRADING consists of the graph G and requests as described previously. We denote by n the number of nodes in V.

For a given graph G and a request (s, t, c) we say that a path P is *valid* if it connects s to t and visits the ASes in the order that is associated with this request by the BGP protocol. This means, in particular, that every valid path goes through every AS at most once. A solution to PATH TRADING is a mapping that maps each request $(s, t, c) \in R$ to a valid path from s to t in graph G. Let us assume that the requests in R are $(s_1, t_1, c_1), \ldots, (s_r, t_r, c_r)$ and that the paths $P_1, \ldots, P_r$ have been chosen for these requests. Then AS i incurs costs on all edges in E_i, i.e., it incurs a total cost of

$$\sum_{j=1}^{r} \Big(c_j \cdot \sum_{e \in P_j \cap E_i} \ell(e) \Big). \tag{1}$$

The *hot-potato route* of a request (s, t, c) is defined to be the concatenation of shortest path routes for the ASes it goes through. To be precise, assume that the BGP protocol associates the route $i_1, \ldots, i_m$ with $s \in V_{i_1}$ and $t \in V_{i_m}$ with this request. Then AS i_1 sends the request from s to the vertex $s_2 \in V_{i_2}$ that is closest to s along the shortest path. Then AS i_2 sends the request from s_2 to the vertex $s_3 \in V_{i_3}$ that is closest to s_2 along the shortest path, and so on. The complete hot-potato route for request (s, t, c) is then the concatenation of these paths. Note that the hot-potato route is not necessarily unique, and in the following we will assume that some hot-potato route is chosen for each request.

Consequently, the costs of the hot-potato routing that an AS i incurs are equal to Equation 1, where the paths $P_1, \ldots, P_r$ are the hot-potato paths. We call a solution to PATH TRADING a *path trade*. Moreover, if the costs for all involved ASes are less than or equal to their hot-potato costs, then we call it a *feasible path trade*. In the following, let $[n]$ be the set of integers $\{1, ..., n\}$. For a vector $x \in \mathbb{R}^n$, let x_i be the i-th component of x.

3. Complexity Results

Our first result is about the complexity of PATH TRADING and extends the weak NP-hardness result of Shavitt and Singer [14]. The proof uses a reduction from 3-PARTITION.

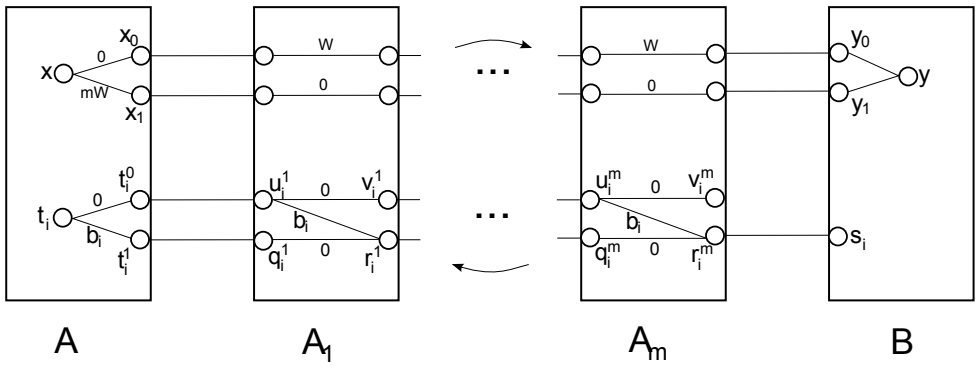

Figure 1: This figure depicts the reduction from 3-Partition to Path Trading in Theorem 1.

Theorem 1. *Finding a feasible path trade apart from the hot-potato routing is strongly NP-hard.*

Proof. We will prove the statement by a reduction from the following version of 3-Partition. An instance of 3-Partition consists of a multiset $S = \{b_1, \ldots, b_{3m}\}$ of $3m$ positive integers, where $\sum_{i=1}^{3m} b_i = mW$ for some integer W and for every i it holds that $W/4 < b_i < W/2$. The problem is to determine whether there exists a partition of S into m subsets of size three such that the elements in every subset add up to W. This problem is strongly NP-hard, i.e. it is NP-hard even if all the numbers b_i are bounded by a polynomial in m.

From an instance for 3-Partition we will construct an instance of Path Trading such that there exists a feasible path trade apart from the hot-potato routing in that instance if and only if there exists a solution to the instance of 3-Partition.

The instance will consist of $m + 2$ ASes A, B, and $A_1, \ldots, A_m$. There is one request $(x, y, 1)$ where $x \in A$ and $y \in B$, which has to be routed through the ASes in the order A, A_1, $\ldots$, A_m, B. In the AS A the node x is connected to two boundary nodes x_0 and x_1, where the weight of the edge xx_0 is 0 and the weight of the edge xx_1 is mW. Moreover, the node y in the AS B is connected to two boundary nodes y_0 and y_1 with edges of weight 0. In addition, there is a unique path that connects x_0 and y_0 going through the ASes $A_1, \ldots, A_m$ and using one edge of weight W in each AS, and there is a unique path that connects x_1 and y_1 going through the ASes $A_1, \ldots, A_m$ and using one edge of weight 0 in each AS.

We note that the request (x, y) can only be routed in two different ways:

under the hot-potato strategy it is routed via x_0 and y_0 and A incurs a cost of 0, while each A_j incurs a cost of W. The second option is to route the request via x_1 and y_1 with a cost of mW for A and a cost of 0 for each A_j.

In addition to the above request, there will be $3m$ additional requests, each corresponding to one element in the multiset S. For each $b_i \in S$ there will be a request $(s_i, t_i, 1)$, where $s_i \in B$ and $t_i \in A$ that has to be routed in the order B, A_m, ..., A_1, A. For each AS A_j, there will be four nodes $q_i^j, r_i^j, u_i^j, v_i^j \in A_j$ together with three edges: $q_i^j r_i^j$ and $u_i^j v_i^j$ of weight 0 and $u_i^j r_i^j$ of weight b_i. For $1 \leq j \leq m-1$, v_i^j is connected to u_i^{j+1} and r_i^j is connected to q_i^{j+1}. Finally, t_i is connected to u_i^1 with an edge of weight 0 via the node t_i^0 and to q_i^1 with an edge of weight b_i via the node t_i^1, and s_i is connected to r_i^m. See Figure 1 for an overview of the nodes, edges, and requests.

Under the hot-potato strategy any request (s_i, t_i) is routed through the nodes $s_i, r_i^m, q_i^m, \ldots, r_i^1, q_i^1, t_i^1, t_i$ for a cost of 0 for each A_j and for a cost of b_i for A. Thus the total cost of all requests (s_i, t_i) for AS A is mW. Each request (s_i, t_i) can also be "rerouted in A_j" by using the edge $r_i^j u_i^j$. This increases the cost of A_j by b_i and decreases the cost of A by b_i when compared to the hot-potato strategy. It is straightforward to see that in any reasonable path trade every request (s_i, t_i) goes through at most one edge $r_i^j u_i^j$ because otherwise the costs for some ASes can be decreased without increasing the costs for other ASes.

In order to finish the proof, we first assume that there exists a solution $S_1, \ldots, S_m$ to the instance of 3-Partition, i.e. $\bigcup_{j=1}^m S_j = S$ and $\sum_{b_i \in S_j} b_i = W$ for every $1 \leq j \leq m$. The corresponding path trading solution that deviates from the hot-potato strategy is to route the request (x, y) via x_1 and y_1 with an increase in cost of mW for A and a decrease in cost of W for each A_j. Moreover, each request (s_i, t_i) is rerouted in that A_j for which $b_i \in S_j$. Thus each A_j saves a cost W from the rerouting of (x, y) and has an increase in cost of $\sum_{b_i \in S_j} b_i = W$. Moreover, since each request (s_i, t_i) is rerouted in some A_j, the AS A will save a total of mW in cost, but has an increase in cost of mW from the rerouting of (x, y). Thus this is a feasible rerouting scheme.

Finally, assume that there exists a feasible rerouting scheme of all requests. If the request (x, y) is rerouted, then certainly there is an increase in cost of mW for A and a decrease in cost of W for each A_j. Since the rerouting scheme is feasible, the only way for A to also save at least mW in cost is to have all requests (s_i, t_i) be routed to t_i via t_i^0, i.e. each (s_i, t_i) has

to be rerouted in some A_j. We let $S_j = \{b_i : b_i \text{ is rerouted in } A_j\}$. Since each A_j has saved a cost of W through the rerouting of (x, y), we have that $\sum_{b_i \in S_j} b_i \leq W$ for each j because otherwise the costs for A_j would be larger than in the hot-potato routing. Due to the premise that $W/4 < b_i < W/2$ for all i the only way to achieve all the reroutings while maintaining feasibility for all the ASes A_j is if $|S_j| = 3$ and $\sum_{b_i \in S_j} b_i = W$ for all j. Thus we get a solution to the instance of 3-Partition in this case. Now, if on the other hand one request (s_i, t_i) is rerouted in some A_j, then A_j has an additional cost of b_i and since the only way for A_j to save cost is to reroute (x, y), we are back in the first case. This finishes the proof. □

Note, that in the above proof the edges of length W, mW, and b_i can be replaced by paths of W, mW, and b_i edges of length 1, respectively, which shows that the version of path trading where edges are only allowed to have length 0 or 1 is also strongly NP-hard. Moreover, as the following corollary shows, the proof can also be adapted in such a way that a 3-Partition exists if and only if a path trade exists that is strictly better than the hot potato routing.

Corollary 1. *It is strongly NP-hard to decide whether there exists a feasible path trade that is for all ASes strictly better than the hot-potato routing.*

Proof. We use the same reduction as in the proof of Theorem 1 and adjust edge lengths as follows: We set $\ell(xx_1) = mW - 1/2$, and for all $i \leq 3m$ and for all $j \leq m$, we set $\ell(r_i^j u_i^j) = b_i - \frac{1}{5}$. Additionally, we set $\ell(y_0 y) = 1$. Any 3-partition still gives rise to a feasible path trade in the same way as in the proof of Theorem 1. In this path trade, AS A reduces its costs by $1/2$ compared to the hot-potato routing, every AS A_j reduces its costs by $3/5$, and B reduces its costs by 1.

To argue that any valid path trade apart from the hot-potato routing gives rise to a 3-partition, observe that in any such path trade, request (x, y) has to be rerouted as otherwise none of the other requests can be rerouted either. So assume that request (x, y) is rerouted. Then every request (s_i, t_i) has to be rerouted in one of the ASes $A_1, \ldots, A_m$ as otherwise, AS A would incur increased costs compared to the hot-potato routing.

Again let $S_j = \{b_i : b_i \text{ is rerouted in } A_j\}$. Since all b_i are positive integers, the condition that $W/4 < b_i$ implies that $b_i - W/4 \geq 1/4$ and hence $W/4 < b_i - 1/5$. As $\sum_{b_i \in S_j}(b_i - 1/5) \leq W$ for each j, this implies that each S_j can contain at most three elements. As there are $3m$ elements in total, this implies that each S_j contains exactly three elements. Since

$\sum_{b_i \in S_j}(b_i - 1/5) = \left(\sum_{b_i \in S_j} b_i\right) - 3/5 \leq W$ and every b_i is integer, we also have that $\sum_{b_i \in S_j} b_i \leq W$. As the total sum of elements is mW, this implies that for every j, we have $\sum_{b_i \in S_j} b_i = W$. □

4. Our Algorithm and Smoothed Analysis

Given the above theorems, in order to develop fast algorithms, we concentrate on path trading between two ASes, and will now present our algorithm for this case. As mentioned before, this algorithm is based on the concept of Pareto efficiency and it enumerates all Pareto-optimal path trades. In the worst case the number of Pareto-optimal solutions can be exponential, but our experiments suggest that on real-world data usually only a few solutions are Pareto-optimal. To give a theoretical explanation for this, we apply the framework of smoothed analysis. The algorithm is a dynamic program that adds the requests one after another, keeping track of the Pareto-optimal path trades of those requests that have already been added.

For a request (s, t, c), a path P from s to t, and $i \in \{1, 2\}$, we denote by $C_i(P)$ the costs incurred by AS i due to routing the request along path P. To keep the notation simple, assume in the following discussion w.l.o.g. that $s \in V_1$ and $t \in V_2$. We denote by $\mathcal{P}(s, t)$ the set of all Pareto-optimal valid paths from s to t. Remember that a path is valid if it starts at s, terminates at t, and does not go back to V_1 after leaving V_1 for the first time. Such a path P belongs to $\mathcal{P}(s, t)$ if there does not exist another valid path that induces strictly lower costs for one AS and not larger costs for the other AS than P. We assume that in the case that there are multiple paths that induce for both ASes exactly the same costs, only one of them is contained in $\mathcal{P}(s, t)$.

Let $P \in \mathcal{P}(s, t)$ be some Pareto-optimal path and let $v \in V_1$ be the *boundary node* at which the path leaves AS 1. Then the subpaths from s to v and from v to t must be shortest paths in AS 1 and AS 2, respectively. Otherwise, P cannot be Pareto-optimal. Hence, the number of Pareto-optimal paths in $\mathcal{P}(s, t)$ is bounded from above by the number of boundary nodes of AS 1 that connect to AS 2. For each pair $s \in V_1$ and $t \in V_2$, the set $\mathcal{P}(s, t)$ can be computed in polynomial time.

Our algorithm first computes the set $\mathcal{P}_1$ of Pareto-optimal path trades for only the first request (s_1, t_1, c_1). This is simply the set $\mathcal{P}(s_1, t_1)$. Based on this, it computes the set $\mathcal{P}_2$ of Pareto-optimal path trades for only the first two requests, and so on. Thus the elements in $\mathcal{P}_i$ are tuples $(P_1, \ldots, P_i)$

where each P_j is a valid path for the jth request. Our algorithm can be implemented using a modification of the Nemhauser/Ullmann algorithm for the knapsack problem [10].

Algorithm 1 Algorithm to compute the Pareto set

$\mathcal{P}_1 = \mathcal{P}(s_1, t_1)$;
for $i = 2$ to r **do**
$\mathcal{P}_i = \{(P_1, \ldots, P_i) \mid (P_1, \ldots, P_{i-1}) \in \mathcal{P}_{i-1}, P_i \in \mathcal{P}(s_i, t_i)\}$;
Remove all solutions from $\mathcal{P}_i$ that are dominated by other solutions from $\mathcal{P}_i$.
If $\mathcal{P}_i$ contains multiple solutions that induce for both ASes exactly the same costs, then remove all but one of them.
end for
return $\mathcal{P}_r$

Theorem 2. *For $i \in [r]$, the set $\mathcal{P}_i$ computed by Algorithm 1 is the set of Pareto-optimal path trades for the first i requests. In particular, the set $\mathcal{P}_r$ is the set of Pareto-optimal path trades for all requests. Algorithm 1 can be implemented to run in time $O(n \log n \cdot \sum_{i=1}^{r} |\mathcal{P}_i| + nr|E| \log n)$*

Proof. Let us first argue that $\mathcal{P}_i$ is the set of Pareto-optimal path trades for the first i requests. For $i = 1$, it follows from the definition that $\mathcal{P}_1 = \mathcal{P}(s_1, t_1)$ is the set of Pareto-optimal path trades for the first request. Now let $i > 2$ and let $(P_1, \ldots, P_i)$ be a Pareto-optimal path trade. We first claim that then P_i must be from the set $\mathcal{P}(s_i, t_i)$. This is because if there was a dominating path $Q \in \mathcal{P}(s_i, t_i)$, then $(P_1, \ldots, P_{i-1}, Q)$ would dominate $(P_1, \ldots, P_i)$, contradicting the assumption that $(P_1, \ldots, P_i)$ is a Pareto-optimal path trade. By exactly the same argument one can argue that $(P_1, \ldots, P_{i-1})$ has to be a Pareto-optimal path trade for the first $i-1$ requests. Hence, after the first step in the for loop of Algorithm 1, the set $\mathcal{P}_i$ is a superset of the set of Pareto-optimal path trades of the first i requests. In the following two steps only dominated solutions are removed and for every cost vector only one solution inducing it is kept. This guarantees that afterwards the set $\mathcal{P}_i$ contains exactly the Pareto-optimal path trades of the first i requests. The most straightforward implementation of the algorithm would yield a running time that is polynomial in n and the cardinalities of $\mathcal{P}_i$, but the degree of this polynomial would be larger than the one claimed. We will argue now that an implementation reminiscent of

the Nemhauser/Ullmann algorithm for the knapsack problem [10] achieves the claimed running time. For this, first of all observe that for each request the set $\mathcal{P}(s_i, t_i)$ can be computed by at most $2n$ shortest path computations as there are at most n boundary nodes. Each of these calculations can be done, e.g., by Dijkstra's algorithm in time $O(|E| \log n)$, yielding a total running time of $O(nr|E| \log n)$ for the shortest path calculations.

Now assume that all sets $\mathcal{P}(s_i, t_i)$ have been computed. In order to speed up the algorithm we will maintain the following invariant: when iteration i of the for loop starts, the set $\mathcal{P}_{i-1}$ is stored in a linked list that is sorted in increasing order of $C_1(P)$. That is, we keep $\mathcal{P}_{i-1}$ sorted according to the costs of AS 1. Now let us look at the first step in the for loop. This for loop creates for each path $P \in \mathcal{P}(s_i, t_i)$ a copy of $\mathcal{P}_{i-1}$ and adds P to all elements of this copy. Hence, if $\mathcal{P}_{i-1}$ is sorted, then without additional effort also these copies are sorted. The total time that this step takes is $O(|\mathcal{P}_{i-1}| \cdot |\mathcal{P}(s_i, t_i)|) = O(n \cdot |\mathcal{P}_{i-1}|)$.

Now we have at most n copies of $\mathcal{P}_{i-1}$ and we have to merge them into a single sorted list. Once this sorted list is computed, duplicates and dominated solutions can be removed by a single linear scan. Merging n sorted lists of length $|\mathcal{P}_{i-1}|$ each can be accomplished in time $O(n \log n \cdot |\mathcal{P}_{i-1}|)$ by standard merging algorithms [7]. Adding up the running times for all the iterations of the for loop yields the claimed bound. □

The above theorem shows that the running time of Algorithm 1 largely depends on the sizes of the Pareto sets. The algorithm of Shavitt and Singer is a dynamic programming algorithm as well, but its running time depends on the range of the possible utility values for both ASes, and this number is in general larger than the size of the Pareto set. In the following we will argue why the Pareto set is small for those instances that we consider. This gives a theoretical justification of our fast running times for Algorithm 1 obtained later in Section 5.

We start by reviewing a result due to Beier et al. [2] who analyzed the number of Pareto-optimal solutions in binary optimization problems with two objective functions. They consider problems whose instances have the following form: the set of feasible solutions S is a subset of $\{0, \ldots, F\}^n$ for some integers F and n, and there are two objective functions $w^{(1)} : S \to \mathbb{R}$ and $w^{(2)} : S \to \mathbb{R}$ that associate with each solution $x \in S$ two weights $w^{(1)}(x)$ and $w^{(2)}(x)$ that are both to be minimized. While $w^{(2)}$ can be an arbitrary function, it is assumed that $w^{(1)}$ is linear of the form $w^{(1)}(x) = w_1 x_1 + ... + w_n x_n$.

In a worst-case analysis, the adversary would be allowed to choose the set of feasible solutions S, and the two objective functions $w^{(1)}$ and $w^{(2)}$. In this setting there are choices S, $w^{(1)}$ and $w^{(2)}$, such that the number of Pareto-optimal solutions is exponential. To make the adversary less powerful and to rule out pathological instances, we assume that the adversary cannot choose the coefficients $w_1, \ldots, w_n$ exactly. Instead he can only specify a probability distribution for each of them according to which it is chosen independently of the other coefficients. Without any restriction, this would include deterministic instances as a special case, but we allow only probability distributions that can be described by a density function that is upper bounded by some parameter $\phi \geq 1$. We denote by $f_i : \mathbb{R}_{\geq 0} \to [0, \phi]$ the probability density according to which w_i is chosen, and we assume that the expected value of w_i is in $[0, 1]$. The parameter ϕ describes the trade-off between the number of Pareto-optimal solutions and the values that the probability distributions can take. This relation is described in the following theorem.

Theorem 3 (Beier et al., [2])**.** *For any choice of feasible solutions $\mathcal{S} \subseteq \{0, ..., F\}^n$, any choice of $w^{(2)}$ and any choice of density functions $f_1, \ldots, f_n : \mathbb{R}_{\geq 0} \to [0, \phi]$, the expected number of Pareto-optimal solutions is bounded by $O(\phi n^2 F^2 \log F)$.*

Now we formulate our problem in terms of Theorem 3. For this, we assume that all requests have positive integer costs. Let F denote an upper bound on the maximal costs possible on any edge, e.g., $F = \sum_{(s,t,c) \in R} c$. Let $m = |E|$ and assume that the edges in E are ordered arbitrarily. Then each path trade leads to a cost vector $x \in \{0, \ldots, F\}^m$ where x_1 denotes the total cost of all requests that use the first edge in E, x_2 denotes the total cost of all requests that use the second edge in E, and so on. If two solutions lead to the same cost vector, then it suffices to remember one of them, and hence, we can assume that the set of possible path trades can essentially be described by the set $S \subseteq \{0, \ldots, F\}^m$ of possible cost vectors. Given such a cost vector $x \in \{0, \ldots, F\}^m$, we can express the cost $w^{(1)}(x)$ of the first AS simply as $\sum_{e \in E_1} \ell(e) x_e$. The costs of the second AS can be defined analogously, and so it looks like that Theorem 3 directly applies when we perturb all edge lengths $\ell(e)$ of edges $e \in E_1$ as these edge lengths are the coefficients in the linear objective function $w^{(1)}$. However, there is a small twist. In Theorem 3 all coefficients in the linear objective function $w^{(1)}$ are chosen randomly. Our objective function, however, does not contain terms for the edges $e \in E_2$. Or with other words the coefficients are 0 for

these edges. If we apply Theorem 3 directly, then also these zero coefficients would be perturbed, which would destroy the combinatorial structure of the problem as then suddenly the cost of the first AS would depend on what happens on edges $e \in E_2$.

To avoid this side effect, we remodel the feasible region S. As argued before, each solution leads to a cost vector $x \in \{0, \dots, F\}^m$, but now we care only about the part of the vector for E_1. Let us define $m' = |E_1| \leq m$. Then each solution leads to a cost vector $x \in \{0, \dots, F\}^{m'}$ that contains only the costs of the first AS. Now of course different solutions can lead to the same vector x if they differ only in the way how the traffic is routed in the second AS. However, Theorem 3 allows completely general objective functions $w^{(2)}$, which we exploit by defining $w^{(2)}(x)$ for a vector $x \in \{0, \dots, F\}^{m'}$ to be the smallest cost for the second AS that can be achieved with any solution whose cost vector for the first AS results in x. This formulation implies the following corollary.

Corollary 2. *Given a path trading instance in which the edge lengths $\ell(e)$ for all $e \in E_1$ are randomly chosen according to probability distributions that satisfy the same restrictions as those in Theorem 3, the expected number of Pareto-optimal solutions is bounded by $O(\phi m^2 F^2 \log F)$.*

Given that the expected number of Pareto-optimal solutions is small, we still have to show that Algorithm 1 computes the Pareto curve in expected polynomial time. This will be established by the following Corollary.

Corollary 3. *Algorithm 1 computes the Pareto curve in expected time $O(\phi n m^2 \cdot \log n \cdot r F^2 \log F)$.*

Proof. The corollary follows immediately from Theorem 2 and Corollary 2 when we observe that Corollary 2 does not only bound the expected number of Pareto-optimal path trades for all requests, but that it also bounds for each i the expected number q_i of Pareto-optimal path trades for requests $1, \dots, i$. Then linearity of expectation implies that the running time is bounded by $O(n \log n \cdot \sum_{i=1}^{r} \mathrm{E}(q_i) + nr|E| \log n) = O(\phi n m^2 \log n \cdot r F^2 \log F)$. □

The reason that we concentrate our efforts on path trading between two ASes was the hardness result in Theorem 1. We can extend this result and also show that there is no hope for Path Trading with an arbitrary number of ASes to be solvable efficiently in the framework of smoothed analysis.

Theorem 4. *There is no smoothed polynomial time algorithm for* Path Trading *with an arbitrary number of ASes, unless* $NP \subseteq BPP$.

Proof sketch. We only present the main idea of the proof. Assume that we have an algorithm A for Path Trading with an arbitrary number of ASes whose running time in the model of smoothed analysis is bounded with constant probability by a polynomial in the input size and the parameter ϕ. We claim that we can utilize this algorithm to solve the instances in Corollary 1 with constant probability in polynomial time. This would immediately imply that 3-Partition is in BPP.

The main idea is that the instances of Path Trading that we construct in Corollary 1 are rather robust against small perturbations of the edge lengths and satisfy the following two properties. If there exists a solution to the given 3-Partition instance, there exists a feasible path trade in which the costs for each AS are smaller by at least some constant c compared to the hot-potato routing. If there does not exist a solution to the given 3-Partition instance, any path trade deviating from the hot-potato routing increases the costs of at least one AS by at least a constant c' and is hence infeasible. Now we consider the smoothed scenario in which we cannot precisely specify the edge lengths anymore, but only choose a probability density for each of them. We choose ϕ polynomially large in m and assume that every edge length is chosen according to a uniform distribution from an interval of length $1/\phi$ around the value that it gets in the reduction in Corollary 1. Due to the aforementioned two properties, an instance generated at random according to these density functions still satisfies the property that there is a feasible path trade that is cheaper for each AS than the hot-potato routing if and only if there exists a solution to the 3-Partition instance if ϕ is sufficiently large. This is because the uncertainty induced by the random choices of the edge lengths is smaller than the constants c and c'.

Hence, we can solve 3-Partition as follows: apply the aforementioned randomized reduction to the given instance, and then run algorithm A for polynomially many steps. If it stops, which happens with constant probability, then we know whether or not there exists a solution to the given 3-Partition instance. Otherwise, we output yes or no with probability 1/2 each. This algorithm outputs the correct solution with probability larger than 1/2. □

5. Computational Results

In this section we present the experimental results about the performance of our algorithm on the IP-level Internet graph from DIMES [13]. We compare it, in particular, with the performance of the dynamic program used by Shavitt and Singer, and we answer the following questions:

- (Running time) *How fast can we compute the Pareto curve?* Algorithm 1 is very fast and scales well.
- (Participation) *How many ASes are involved in path trading?* In our experiments we see that almost all ASes can benefit from path trading.
- (Gain) *How much do ASes gain by trading?* The gains are typically modest, around 5%.

The answers to these questions are, of course, depending on the assumptions we made. As Shavitt and Singer, we assume that traffic is symmetric, i.e., the number of requests sent from AS A to AS B is the same as the number of requests sent from AS B to AS A for every pair of ASes. This assumption is not necessarily true, but it is common and used, e.g., in [14], [9], and [15]. One could imagine that in real networks requests are not evenly spread, but are more concentrated between popular ASes for example. However, we can still see from our experiments that ASes have a good chance of gaining from path trading as long as there is non-zero traffic in both directions. Our second assumption is that each request between two ASes has to be routed between two nodes that are chosen uniformly at random from these ASes. Our third assumption is that all edges have length 1, i.e., the number of hops is used to measure the costs of an AS for routing a request. By absence of real data, we feel that this is a reasonable and common assumption.

In the following subsection we present the details of the experimental setup. The algorithm by Shavitt and Singer is named Algorithm 2. Then we show and discuss the experimental results.

5.1. Experimental Setup

The test set-up for answering the questions differs slightly, and experiments were performed on different data sets from DIMES [13].

Experimental set-up for "Running time". In order to determine AS pairs where there is potential for path trading and therefore a larger set of Pareto-optimal solutions, we simulated a small number of requests for each sufficiently connected pair, i.e. pairs for which there are at least two ingress/egress nodes between the two networks (otherwise at least one network has a unique path to route all requests).

To measure how many ASes could benefit from path trading, we simulated 5 requests for each of the 4348 sufficiently connected AS pairs in either direction. For comparing performance we selected a subset of 15 AS pairs arbitrarily among the AS pairs that benefited from path trading.

For both algorithms, we need to calculate shortest paths beforehand. Because of the large number of possible routings, many shortest paths need to be computed. This was all done as part of the preprocessing, and all shortest paths were stored in a hash-table for fast access for both algorithms. In the following, this time is not included in the running times of the algorithms.

Experimental set-up for "Participation" and "Gain". The data set is very large. Therefore, computing *for all* pairs of ASes how much could be gained from path trading has some bottlenecks. The first bottleneck is the computation of shortest paths, since for large graphs the standard algorithms such as Dijkstra run out of memory or are too slow. Since the development of fast algorithms for shortest paths in Internet graphs is out of scope of this paper, we used some other techniques to make this calculation feasible.

Consider the network of a single AS. We call a node a *border node* if it shares edges with nodes outside its AS. For every pair of ASes, we select 100 nodes uniformly at random that are in both ASes and requests from an AS to another AS can only go through these nodes. Secondly, from all other border nodes we select 1000 nodes uniformly at random where requests can originate from. These are reasonable restrictions since the goal of the experiment is to see how path trading performs on real-life topologies of ASes.

Figure 2 summarizes the characteristics of the network for these experiments.

5.2. Experimental Results

Running time

Table 1 shows a comparison of the running times of our algorithm and the algorithm of Shavitt and Singer. The running times are the total of

Size

Low	***Up***	**# ASes**
0	0	77
1	2	2210
3	5	4769
6	10	4123
11	25	4745
26	100	5590
101	200	1981
201	500	1635
501	1000	654
1001	2000	383
2001+		420

(a) The number of nodes in each AS, where **# ASes** indicates how many ASes have at least ***Low*** and at most ***Up*** nodes.

Border size

Low	***Up***	**# ASes**
0	0	897
1	2	4779
3	5	5456
6	10	4253
11	25	4506
26	100	4209
101	200	1084
201	500	760
501	1000	317
1001	2000	165
2001+		161

(b) The number of outgoing edges from ASes, where **# ASes** indicates how many ASes have at least ***Low*** and at most ***Up*** outgoing edges.

Shared border size

Low	***Up***	**# AS pairs**
0	0	872
1	2	32525
3	5	16188
6	10	10064
11	25	10052
26	100	8362
101	200	1656
201	500	1073
501	1000	407
1001	2000	177
2001+		96

(c) The number of shared edges betwen ASes, where **# AS pairs** indicates how many pairs of ASes have at least ***Low*** and at most ***Up*** shared edges.

Figure 2: The network characteristics of the internet graph used in the experiments for "Gain" and "Participation"

the running time over the 15 selected pairs in seconds. As can be seen, the running time of the algorithm of Shavitt and Singer quickly becomes very high with an increasing number of requests. This confirms our observation about the running times after the proof of Theorem 2.

# Requests	**Our Algorithm (s)**	**Shavitt's and Singer's Algorithm (s)**	**Ratio :**
1	0.02	0.09	1: 4.50
5	0.19	6.04	1: 31.79
10	1.09	84.31	1: 77.35
15	2.38	270.87	1: 113.81
19	4.01	519.27	1: 129.49

Table 1: The performance of Algorithm 1 compared to Algorithm 2.

The memory usage is dominated by the number of Pareto optimal solutions, and each Pareto optimal solution is represented as a tuple of two integers. Figure 3 shows a graphical comparison of both algorithms. Not only is our algorithm fast for few requests, it can handle up to ten times more requests in the same time as the algorithm by Shavitt and Singer.

Participation and Gain

From Figure 4 we see that roughly 90% of all AS pairs profit from trading paths. While there are many assumptions underlying these experiments that are not neccesarily realistic, this indicates that many ASes would benefit from path trading in practice. The gains are modest, typically in the range of $0 - 10\%$ and a small percentage has significantly higher gains, i.e. more than 15%. Even when the number of possible ways to route to another AS is limited, the gain is in the same range.

The running times are, again, very fast and the real bottleneck is the calculation of shortest paths. In all cases computation time is at most *seconds* discounting the shortest path computations.

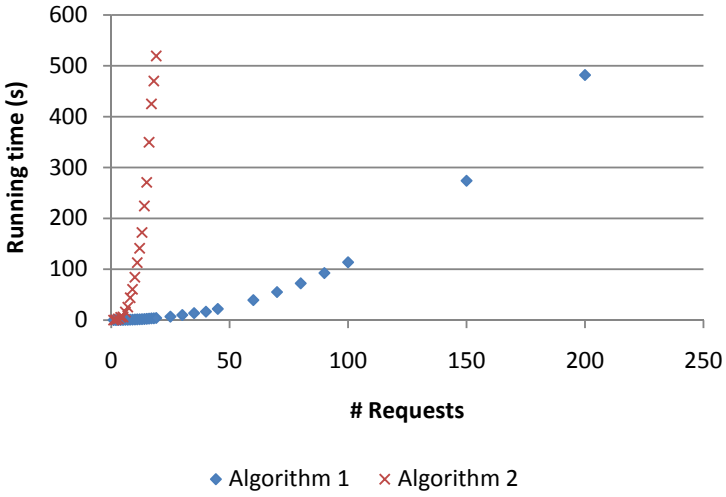

Figure 3: Running times of both algorithms.

Gain	**# AS pairs**	**%**
0%	2601	9.39%
$0-4\%$	8096	29.23%
$4-10\%$	11933	43.08%
$10-15\%$	3584	12.94%
$15-20\%$	1009	3.64%
20%+	478	1.73%

Figure 4: The number and percentage of AS pairs that had a certain amount of gain due to path trades. **# AS pairs** and % are the number of AS pairs and the percentage of AS pairs, respectively that reduced costs by ***Gain***.

	Shared border size			
	2	***26***	***101***	***201+***
Gain	***25***	***100***	***200***	
0%	2126	390	32	53
$0-4\%$	5707	1776	295	318
$4-10\%$	6751	3586	756	840
$10-15\%$	1796	1171	283	334
$15-20\%$	464	359	92	94
20%+	270	151	33	24

Figure 5: The number of AS pairs that had a certain amount of gain due to path trades split by the number of shared edges. An entry of the table is the number of AS pairs that reduced costs by ***Gain*** and that shared ***Shared border size*** many edges.

6. Conclusions

Despite that Path Trading is NP-hard, we developed an algorithm that is fast in practice, which we explained theoretically by smoothed analysis. Our algorithm is robust, and gives great flexibility because it returns the whole set of Pareto-optimal path trades.

Throughout the paper we assumed that ASes are *truthful* and share their data. The real costs can, of course, be scaled to the interval $[0, 1]$, where 1 are the costs of the hot-potato routing [9]. This would keep the real costs secret, but we would have a trade-off of relative gain and not absolute cost savings. Lying could potentially increase the cost savings of an AS, but no AS would be worse off than with hot-potato routing. Every AS simply refuses any path trade that costs more than their hot-potato strategy. Unfortunately, achieving a mechanism that is both truthful and efficient seems impossible [14].

The most promising direction for further research is to use real traffic data, and model this in an appropriate way to find out how large the real gains are for ASes. From a theoretical perspective, extending Corollary 2 from two ASes to a fixed number of ASes would be interesting. There is a generalization of the results by Beier et al. [2] to multiobjective optimization problems with any constant number of objectives by Brunsch and Röglin [4] that can even handle perturbations in which some coefficients can be deterministically set to zero. However, this generalization cannot be applied directly to path trading with multiple ASes because only binary solution sets $S \subseteq \{0, 1\}^n$ are considered.

Acknowledgements.. The authors would like to thank Tobias Brunsch for proofreading this manuscript and for his helpful comments.

[1] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A measurement-based analysis of multihoming. In *ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages pages 353–364, 2003.

[2] R. Beier, H. Röglin, and B. Vöcking. The smoothed number of pareto optimal solutions in bicriteria integer optimization. In *Integer Programming and Combinatorial Optimization, 12th International Conference (IPCO)*, pages 53–67, 2007.

[3] André Berger, Heiko Röglin, and Ruben van der Zwaan. Path trading: Fast algorithms, smoothed analysis, and hardness results. In *SEA*, pages 43–53, 2011.

[4] T. Brunsch and H. Röglin. Improved smoothed analysis of multiobjective optimization. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 407–426, New York, NY, USA, 2012. ACM.

[5] R. Dai, D. O. Stahl, and A. B. Whinston. The economics of smart routing and qos. In *Group Communications and Charges; Technology and Business Models, 5th COST264 International Workshop on Networked Group Communications (NGC)*, pages 318–331, 2003.

[6] D.K. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In *ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 79–82, 2004.

[7] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition*. Addison-Wesley, 1997.

[8] Y. Liu and A.L.N. Reddy. Multihoming route control among a group of multihomed stub networks. *Computer Communications*, 30(17):3335–3345, 2007.

[9] R. Mahajan, D. Wetherall, and T. Anderson. Negotiation-based routing between neighboring isps. In *2nd Symposium on Networked Systems Design and Implementation (NSDI)*, pages 29–42, 2005.

[10] G.L. Nemhauser and Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.

[11] B. Quoitin and O. Bonaventure. A cooperative approach to interdomain traffic engineering. In *In Proceedings of EuroNGI*, 2005.

[12] P. Sevcik and J. Bartlett. Improving user experience with route control. *Technical Report NetForecast Report 5062, NetForecast, Inc.*, 2002.

[13] Y. Shavitt and E. Shir. DIMES: let the internet measure itself. *ACM SIGCOMM Computer Communication Review*, 35(5):71–74, 2005.

[14] Yuval Shavitt and Yaron Singer. Limitations and possibilities of path trading between autonomous systems. In *INFOCOM*, pages 1226–1234, 2010.

[15] G. Shrimali, A. Akella, and A. Mutapcic. Cooperative interdomain traffic engineering using nash bargaining and decomposition. In *26th IEEE International Conference on Computer Communications (INFOCOM)*, pages 330–338, 2007.

[16] D. A. Spielman and S.-H. Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52(10):76–84, 2009.

[17] D.A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[18] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in ip networks. In *International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS)*, pages 307–319, 2004.

[19] J. Winick, S. Jamin, and J. Rexford. Traffic engineering between neighboring domains. *Technical Report*, 2002.

[20] Y.R. Yang, H. Wang H. Xie, A. Silberschatz, A. Krishnamurthy, Y. Liu, and E.L. Li. On route selection for interdomain traffic engineering. *IEEE Network*, 19(6):20–27, 2005.