Discrete Applied Mathematics 185 (2015) 8–17

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

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

ARTICLE INFO

Article history:
Received 21 June 2013
Received in revised form 13 November 2014
Accepted 25 November 2014
Available online 18 December 2014

Keywords:
Shortest paths
Internet routing
Smoothed analysis

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.

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 behavior 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

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

* Corresponding author. Tel.: +31 433884894; fax: +31 433882000.
E-mail addresses: a.berger@maastrichtuniversity.nl (A. Berger), roeglin@cs.uni-bonn.de (H. Röglin), r.vanderzwaan@maastrichtuniversity.nl (R. van der Zwaan).

http://dx.doi.org/10.1016/j.dam.2014.11.025
0166-218X/

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 [16] 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 [17] 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 $V_1, \dots, V_k$, 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 $E_1, \dots, E_k$, 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