

Cuts and Disjoint Paths in the Valley-Free Path Model of Internet BGP Routing^{*}

Thomas Erlebach¹, Alexander Hall^{2,**}, Alessandro Panconesi³, and Danica Vukadinović^{2,**}

¹ Dept. of Computer Science, University of Leicester, Leicester LE1 7RH, UK,
`t.erlebach@mcs.le.ac.uk`

² Computer Engineering and Networks Laboratory (TIK), Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland,
`{hall|vukadin}@tik.ee.ethz.ch`

³ DSI – Università La Sapienza, Rome, Italy,
`ale@dsi.uniroma1.it`


Abstract. In the valley-free path model, a path in a given directed graph is valid if it consists of a sequence of forward edges followed by a sequence of backward edges. This model is motivated by BGP routing policies of autonomous systems in the Internet. Robustness considerations lead to the problem of computing a maximum number of disjoint paths between two nodes, and the minimum size of a cut that separates them. We study these problems in the valley-free path model. For the problem of computing a maximum number of edge- or vertex-disjoint valid paths between two given vertices s and t , we give a 2-approximation algorithm and show that no better approximation ratio is possible unless $P = NP$. For the problem of computing a minimum vertex cut that separates s and t with respect to all valid paths, we give a 2-approximation algorithm and prove that the problem is *APX*-hard. The corresponding problem for edge cuts is shown to be polynomial-time solvable. We present additional results for acyclic graphs.

1 Introduction

Let $G = (V, E)$ be a directed graph. For $s, t \in V$, a path from s to t is *valid* if it consists of a (possibly empty) sequence of forward edges followed by a (possibly empty) sequence of backward edges. Note that a valid path from s to t gives also a valid path from t to s and vice versa. We refer to this model of valid paths as the *valley-free path model*. The reason for this terminology is that if we view directed edges as “pointing upward” towards their heads, a path is valid if and

^{*} Research partially supported by the European Commission in the 5th Framework Programme under contract IST-2001-32007 (APPOL II) and in the 6th Framework Programme under contract 001907 (DELIS), with funding for the Swiss participants provided by the Swiss Federal Office for Education and Science.

^{**} Supported in DICS-Project No. 1838 by the Hasler Foundation.

only if it does not contain a “downward” edge followed by an “upward” edge, i.e., a valley ().

The motivation for studying the valley-free path model comes from BGP routing policies in the Internet on the level of autonomous systems, as explained in Section 1.1 in more detail. Robustness considerations of the Internet topology then lead naturally to questions concerning the computation of large sets of disjoint valid paths between two given vertices, and of small vertex or edge cuts separating two given vertices with respect to all valid paths. The corresponding optimization problems for standard directed paths can be solved efficiently using network flow techniques (e.g., see [1]). In this paper, we initiate the investigation of these problems in the valley-free path model. It turns out that several of these problems are *NP*-hard in this model. Our main results are:

- We give 2-approximation algorithms for the problems of computing a maximum number of vertex- or edge-disjoint valid paths between two given vertices s and t , and we show that it is *NP*-hard to approximate these problems within ratio $2 - \varepsilon$ for any fixed $\varepsilon > 0$.
- We prove *APX*-hardness for the problem of computing a min valid s - t -vertex-cut, i.e., a minimum size set of vertices whose removal from G disconnects all valid paths between s and t . We also give a 2-approximation algorithm for this problem.
- For the edge version of the problem, i.e., computing a min valid s - t -edge-cut, we give a polynomial algorithm that computes an optimal solution.
- We prove that the size of a min valid s - t -cut is at most twice the maximum number of disjoint valid s - t -paths, both for the edge version and the vertex version of the problems, and we show that this bound is tight.
- For the special case that the given graph G is acyclic (where “acyclic” is to be understood in the standard sense, i.e., the directed graph G is acyclic if it does not contain a directed cycle), we obtain a polynomial algorithm for finding k edge- or vertex-disjoint valid paths between s and t if they exist, where k is an arbitrary constant. The algorithm is based on a generalization of ideas due to Fortune, Hopcroft and Wyllie [9]. We also establish *NP*-hardness for the general problem of computing a maximum number of vertex- or edge-disjoint valid s - t -paths in acyclic graphs.

Our results provide interesting insights into natural variations of the classical problems of computing disjoint s - t -paths and minimum s - t -cuts. Furthermore, the algorithms we provide may be useful for investigating issues related to the robustness of the Internet topology while taking into account the effects of routing policies.

1.1 Motivation: Autonomous Systems on the Internet

In this section we discuss the issues in Internet routing on the autonomous system level that have motivated our study. An autonomous system (AS) on the Internet is a subnetwork under separate administrative control. ASs are

connected by physical links and exchange routing information using the Border Gateway Protocol (BGP). An AS can consist of tens to thousands of routers and hosts. On the level of ASs, the Internet can be represented as an undirected graph with a vertex for each AS and an edge between two ASs if they have at least one physical link between them. But such an undirected graph is not sufficient to model the effects of routing policies enforced by individual ASs. Each AS announces the routes to a certain set of destination ASs (more precisely, address prefixes) to some of its neighbors. The decisions which routes to announce to which neighbors are determined by BGP routing policies. These depend mostly on the economic relationships between ASs and represent an important aspect of the Internet structure.

The nature of commercial agreements between ASs has attracted a lot of attention in the Internet economics research community [12, 13, 3]. The main trends in the diversity of these agreements were described in [12, 13]. The impact of economic relationships on the engineering level, more precisely on BGP routing, has not been immediately recognized despite the direct implication that an existing link between two ASs will not be used to transfer traffic that collides with their mutual agreement. Then several papers showing the impact of BGP policies on features such as path inflation and routing convergence have appeared [16, 17, 14].

Consequently, the previously developed undirected model for the AS topology is not satisfactory because it allows some prohibited paths between ASs and thus might produce a distorted picture of BGP routing. On the other hand, involving all of the peculiarities of the contracts between autonomous systems in a new model would add too much complexity. Thus, a rough classification into a few categories was proposed for the BGP policies adopted by a pair of ASs: customer-provider, peer-to-peer, and siblings (see [10]). Later on, a simplified model with only two categories (customer-provider and peer-to-peer) was proposed [15].

A customer-provider relationship between A and B can be represented as a directed edge from A to B, and a peer-to-peer relationship as an undirected edge. If ASs A and B are in a customer-provider relationship, B announces all its routes to A, but A announces to B only its own routes and routes of its own customers. If they are peers, they exchange their own routes and routes of their customers, but not routes that they learn from their providers or other peers. This leads to the model proposed in [15] that a path is valid if and only if it consists of a sequence of customer-provider edges ($\bullet \rightarrow \bullet$), followed by at most one peer-to-peer edge ($\bullet - \bullet$), followed by a sequence of provider-customer edges ($\bullet \leftarrow \bullet$).

It is easy to see that a peer-to-peer edge (undirected edge) between A and B can be replaced by two customer-provider edges from A to X and from B to X, where X is a new node, without affecting the solutions to any of the optimization problems (minimum cut problems and maximum disjoint paths problems) we study in this paper. Therefore, without losing generality, we can consider a model with only customer-provider edges. In other words, this model consists of a directed graph with ASs as nodes and where the edge-directions

represent economic relationships. Here the valley-free paths are exactly the paths permitted by the BGP routing policies.

Information about the economic relationships between autonomous systems is not publicly available. Therefore, several approaches to inferring these relationships from available topology data or AS path information have been proposed in the literature [10, 15, 8, 6].

If a communication network is represented as an undirected or directed graph in a model without routing policies, it is natural to measure the connectivity provided to an s - t -pair as the maximum number of disjoint s - t -paths or the minimum size of an s - t -cut (by Menger’s theorem, these two quantities are the same). This motivates us to study the corresponding notions for the valley-free path model in this paper.

It seems natural to expect that the directed graph of customer-provider edges is acyclic (i.e., does not contain a directed cycle), because providers should be “higher” in the Internet hierarchy than their customers. But it turns out that the graphs obtained with several of the abovementioned algorithms do contain directed cycles. Thus, we are interested in cuts and disjoint paths both in general directed graphs and in acyclic graphs.

1.2 Outline

In Section 2, we give the necessary definitions and discuss some preliminaries. Section 3 contains our results for disjoint paths and minimum cuts in general directed graphs. In Section 4, we consider acyclic graphs. We give our conclusions and point to some open problems in Section 5. Proofs omitted due to space restrictions can be found in [7].

2 Preliminaries

Following the terminology from [15, 6, 8], where the problem of classifying the relationships between ASs is called the Type-of-Relationship (ToR) problem, we call a simple directed graph $G = (V, E)$ a *ToR graph* if G has no loops and no anti-parallel edges (i.e., $(u, v) \in E$ implies $(v, u) \notin E$). In terms of the underlying motivation, a directed edge from u to v , where $u, v \in V$, means that u is a customer of v . A path $p = v_1, v_2, \dots, v_r$ in a ToR graph is *valid* (and called a valid v_1 - v_r -path), if it satisfies the following condition:

There exists some j , $1 \leq j \leq r$, such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq j-1$ and $(v_i, v_{i-1}) \in E$ for $j+1 \leq i \leq r$.

The part of the path from v_1 to v_j is called its *forward part*, the part from v_j to v_r its *backward part*. Note that valid paths are symmetric, i.e., the reverse of a valid s - t -path is a valid t - s -path. The existence of a valid s - t -path can be checked in linear time by performing a standard directed depth-first search from s and from t and testing if any vertex is reachable from both s and t along a directed path.

Let $s, t \in V$ be two distinct vertices in a ToR graph $G = (V, E)$. A set $C \subseteq V \setminus \{s, t\}$ is a *valid s - t -vertex-cut* if there is no valid path from s to t in $G - C$. A smallest such set C is called a *min valid s - t -vertex-cut*. The *min valid s - t -edge-cut* is defined analogously. Two valid s - t -paths are called vertex-disjoint (edge-disjoint) if the only vertices that they have in common are s and t (if they have no edges in common). The optimization problems that we are interested in are those of computing minimum size cuts and maximum size sets of disjoint paths, both in the vertex version and in the edge version: the min valid s - t -vertex-cut problem, the min valid s - t -edge-cut problem, the max vertex-disjoint valid s - t -paths problem, and the max edge-disjoint valid s - t -paths problem. An approximation algorithm A for an optimization problem Π is a polynomial algorithm that always outputs a feasible solution. A is a ρ -approximation algorithm (has approximation ratio ρ), if for all inputs I , $OPT(I)/A(I) \leq \rho$, if Π is a maximization problem, or $A(I)/OPT(I) \leq \rho$, if Π is a minimization problem. Here, $OPT(I)$ is the objective value of an optimal solution, and $A(I)$ is the objective value of the solution computed by A , for a given input I . *APX* is the class of all optimization problems (with some natural restrictions [2]) that can be approximated within a constant factor. A problem is *APX-hard* if every problem in *APX* can be reduced to it via an approximation preserving reduction. For every *APX-hard* problem there is a constant $\rho > 1$ such that it is not possible to find a ρ -approximation algorithm for the problem unless $P = NP$. See [2] for further information about approximability classes and approximation preserving reductions.

3 Results for General ToR Graphs

First, we introduce a helpful *two-layer model* that leads to a relaxation of flows and cuts in ToR graphs.

3.1 The Two-Layer Model

From a ToR graph $G = (V, E)$ and $s, t \in V$ we construct a *two-layer model* H , which is a directed graph, in the following way. Two copies of the graph G are made, called the *lower* and the *upper layer*. In the upper layer all edge-directions are reversed. Every node v in the lower layer is connected with an edge to the corresponding copy of v , denoted v' , in the upper layer. The edge is directed from v to v' . (When dealing with edge-cuts or edge-disjoint paths, we let H contain $|V|$ parallel copies of the edge (v, v') to ensure that these “vertical” edges are not contained in minimum edge-cuts and do not restrict the number of edge-disjoint paths switching from the lower to the upper layer at v .) Finally, we obtain the two-layer model H by identifying the two s -nodes (of lower and upper layer) and also the two t -nodes, and by removing the incoming edges of s and the outgoing edges of t .

A valid path $p = v_1, \dots, v_r$ in G with $v_1 = s$ and $v_r = t$ is equivalent to a directed path in H in the following way. The forward part of p , i.e., all edges

$(v_i, v_{i+1}) \in p$ that are directed from v_i to v_{i+1} , is routed in the lower layer. Then there is a possible switch to the upper layer with a (v, v') type edge (there can be at most one such switch). The backward part of p is routed in the upper layer. If there is only a forward respectively a backward part of p , then the corresponding path in H is only in the lower respectively upper layer. See Fig. 1 for an example.

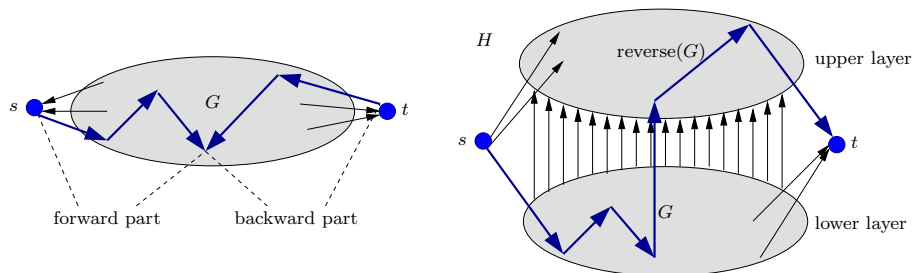


Fig. 1. Path in ToR graph G and corresponding path in the two-layer model H

We now detail in which sense the two-layer model yields relaxations of vertex- respectively edge-disjoint paths and vertex- respectively edge-cuts in ToR graphs.

Note that two vertex-disjoint valid paths in G directly give two vertex-disjoint paths in H . But two vertex-disjoint paths p_1, p_2 in H do not necessarily correspond to vertex-disjoint valid paths in G . The path p_1 might use the node v and the path p_2 its counterpart v' in the other layer, yielding two valid paths that are not vertex-disjoint in G . The analogous statements apply to edge-disjoint paths.

A valid s - t -vertex-cut in G directly gives an s - t -vertex-cut in H of twice the cardinality: simply take for each cut node in G the corresponding nodes from both layers in H . But, of course, there might be an s - t -vertex-cut in H without the property that for each node v in the cut, also its counterpart v' is in the cut. Analogous statements apply to edge-cuts.

3.2 Min Valid s - t -Vertex-Cut

First, we are able to establish the hardness of the min valid s - t -vertex-cut problem by a reduction from the 3-way edge cut problem in undirected graphs.

Theorem 1. *For a given ToR graph $G = (V, E)$ and $s, t \in V$, finding the min valid s - t -vertex-cut is NP-hard and even APX-hard.*

Proof. We use a similar technique as in [11], reducing the undirected 3-way edge cut problem to the min valid s - t -vertex-cut problem in ToR graphs. In the undirected 3-way edge cut problem, we are given an undirected graph G , and three terminals v_1, v_2, v_3 . The goal is to find a minimum set of edges in G such that after removing this set, all pairs of vertices in $\{v_1, v_2, v_3\}$ are disconnected. This problem is proven to be NP-hard in [5].

Let $G = (V, E)$ be such an undirected graph with 3 distinct terminals v_1 , v_2 and v_3 . We create a ToR graph G' in the following way: each node v of G is replaced with $\deg(v)$ copies of the same node. For each edge $\{u, w\}$ in G , a gadget consisting of 2 new nodes, $e_1^{u,w}$ and $e_2^{u,w}$, is added. The gadget includes an edge from $e_1^{u,w}$ to $e_2^{u,w}$, edges from all copies of u and w to $e_1^{u,w}$ and from $e_2^{u,w}$ to all copies of u and w . We also add two nodes s and t and the edges from s to all copies of v_1 , from all copies of v_2 to s and t , and from t to all copies of v_3 . See Fig. 2 for a simple example.

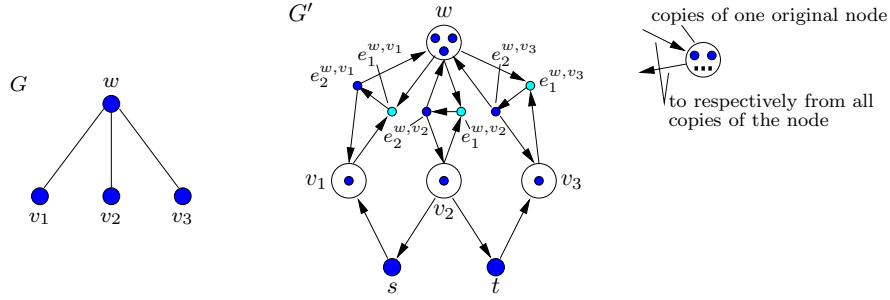


Fig. 2. Example of the transformation of the original undirected graph G to the ToR graph G'

Note that every valid path between any copy of u and any copy of w via the gadget added for the edge $\{u, w\}$ contains $e_1^{u,w}$. This holds because no path “copy of w , $e_2^{u,w}$, copy of u ” or “copy of u , $e_2^{u,w}$, copy of w ” is valid.

In the following we will first show that any valid s - t -vertex-cut in G' can be transformed into a cut of at most the same cardinality that only contains e_1 type nodes. Then we prove that there is a direct correspondence between 3-way edge cuts in G and valid s - t -vertex-cuts in G' that contain only such e_1 type nodes. This yields in particular that an approximation algorithm for the min valid s - t -vertex-cut problem gives an approximation algorithm of the same ratio for the 3-way edge cut problem.

Assume we are given a valid s - t -vertex-cut C in G' that contains nodes that are not of type e_1 . We can assume that C is a minimal cut (inclusion-wise). If the cut contains a copy u' of u , where u is a node in the original graph G , it must also contain all other copies of u . Otherwise there is always an equivalent “detour” path via one of these other copies, rendering the addition of u' to C superfluous and contradicting the minimality of C . If C contains all $\deg(u)$ copies of a node u , we replace these nodes in C by the e_1 type nodes of the neighboring gadgets. There are exactly $\deg(u)$ such nodes. Every valid s - t path containing a copy of u traverses at least one of these neighboring gadgets (and therefore its e_1 node, see above). To see this, note that the paths “ s , copy of v_2 , t ” and “ t , copy of v_2 , s ” are not valid. Thus by this replacement we did not reintroduce previously cut paths. The cardinality of C did not increase. If C contains an e_2 type node, we replace it by the corresponding e_1 type node. Once more the

cardinality of C does not increase and no valid paths are reintroduced. Now C contains only e_1 type nodes.

Next, we show that for a set of edges Q in the graph G , the corresponding set of nodes $C = \{e_1^{u,w} \mid \{u, w\} \in Q\}$ in G' is a valid s - t -vertex-cut if and only if Q is a 3-way cut for terminals v_1, v_2 and v_3 in G . (As previously mentioned, a 3-way cut disconnects all possible pairs in $\{v_1, v_2, v_3\}$.)

Note that the gadgets ensure that for any two nodes u and w in G' corresponding to the endpoints of an undirected edge $\{u, w\}$ in G , there exists a directed path from u to w and from w to u in the gadget added for $\{u, w\}$. To disconnect all such u - w paths, it suffices to cut the $e_1^{u,w}$ node.

First, if C is a valid s - t -vertex-cut, Q is a 3-way cut, because otherwise, if any pair of terminals v_1, v_2, v_3 are connected in G , then there will be a valid path between s and t which will give a contradiction. On the other hand, if Q is a 3-way cut, there is no valid path between s and t in $G' - C$, because at least one gadget is disconnected on every valid path corresponding to an undirected path between v_i and v_j for $i \neq j$ in G and, as noted before, the paths “ s , copy of v_2 , t ” and “ t , copy of v_2 , s ” are not valid.

Thus, we have shown that min valid s - t -vertex-cut is *NP*-hard. *APX*-hardness also follows directly from the *APX*-hardness of 3-way edge cut [5]. \square

Note that the proof does not carry over to min valid s - t -edge-cut, because no gadget can be found where the role of the $e_1^{u,w}$ node is taken over by exactly one edge (such that the copies of u and the copies of w are disconnected if this edge is deleted). In fact, in Section 3.3 we will show that a polynomial-time optimal algorithm exists for the min valid s - t -edge-cut problem.

A Simple 2-Approximation Given a ToR graph $G = (V, E)$ and $s, t \in V$ (where we assume that there is no direct edge in G between s and t , because otherwise a valid s - t -vertex-cut does not exist), the min valid s - t -vertex-cut approximation algorithm is as follows:

ALGORITHM VERTEXCUT

1. From G construct the two-layer model H as described in Section 3.1.
 2. Compute a min s - t -vertex-cut C_H in H .
 3. Output the set $C_G = \{v \in V \mid \text{at least one copy of } v \text{ is in } C_H\}$ as valid s - t -vertex-cut.
-

Clearly $|C_G| \leq |C_H|$ holds and C_G is a valid s - t -vertex-cut in G . Let C_{opt} be a min valid s - t -vertex-cut in G . As mentioned in Section 3.1, by duplicating C_{opt} for both layers of H one obtains an s - t -vertex-cut in H . Thus $|C_H|$ is at most twice $|C_{opt}|$, which gives Theorem 2.

Theorem 2. *There is a 2-approximation algorithm for the min valid s - t -vertex-cut problem in ToR graphs.*

3.3 Min Valid s - t -Edge-Cut

Quite surprisingly, there is a polynomial-time optimal algorithm for the min valid s - t -edge-cut problem in ToR graphs. This is in contrast to many flow and cut problems in directed and undirected graphs, where the node and the edge variant of the respective problem are of the same complexity.

Let `EDGE CUT` be the reformulation of algorithm `VERTEX CUT` from Section 3.2 that considers edges instead of vertices. (Recall that for the edge version of the cut problem, we also modify the 2-layer model H in such a way that each vertex in the lower layer is connected to its copy in the upper layer by $|V|$ parallel edges; this ensures that a minimum edge-cut in H does not contain any edges connecting the lower layer to the upper layer.) The same simple argumentation as above yields that `EDGE CUT` has approximation ratio at most 2 for the min valid s - t -edge-cut problem. The proof of the following theorem, however, shows that this algorithm in fact computes an optimal solution.

Theorem 3. *There is a polynomial-time optimal algorithm for the min valid s - t -edge-cut problem in ToR graphs.*

Proof. We begin by proving a lemma that states a crucial property of (optimal) valid s - t -edge-cuts in ToR graphs.

Lemma 1. *Let $G = (V_G, E_G)$ be a ToR graph, $s, t \in V_G$ and C_G any valid s - t -edge-cut in G . From C_G an s - t -edge-cut C_H in the corresponding two-layer model $H = (V_H, E_H)$ can be derived with $|C_H| = |C_G|$.*

Proof. We start by adding for each edge $e \in C_G$ the two corresponding edges from the lower and upper layer to C_H . This yields an s - t -edge-cut C_H in H with $|C_H| = 2 \cdot |C_G|$, as already described in Section 3.1. Then, iteratively for each edge pair $e, e' \in C_H$ either e or e' is removed from C_H , where $e = (v, w) \in E_H$ is in the lower layer and $e' = (w', v') \in E_H$ is its counterpart in the upper layer. Below we show that assuming C_H is a cut before the removal it will still be a cut afterwards, if the edge is properly chosen. Thus, in the end, after considering all edge pairs in the original cut, C_H is still a cut and $|C_H| = |C_G|$ holds.

Now we consider a single step of the iteration where the pair $e = (v, w), e' = (w', v') \in C_H$ is treated, assuming that the (perhaps already modified) set C_H is still an s - t -edge-cut in H . Assume an s - t -path p exists that traverses only e and no other edge of the cut, i.e. $e \in p$ and $e_c \notin p$, for all $e_c \in C_H \setminus \{e\}$. We claim that in this case no s - t -path p' exists that traverses only e' and no other edge of the cut. It is then safe to remove e' from C_H . Symmetrically, if such a path p' exists, there cannot be a path p and thus e can be removed safely. (If neither p nor p' exists, remove e and continue the iteration.)

Aiming for a contradiction, we assume that both such paths p and p' exist. The edge $e \in p$ is directed from v to w , thus p has the form $s \cdots v w \cdots t$. Let $p_1 = s \cdots v$ denote the first part of p . Analogously the edge $e' \in p'$ is directed from w' to v' and thus p' has the form $s \cdots w' v' \cdots t$. Let $p_2 = v' \cdots t$ denote the last part of p' . Neither p_1 nor p_2 contain an edge from C_H . Therefore, p_1 and p_2 can be recombined via the edge (v, v') to form an s - t -path that does not

contain any cut edge. This is a contradiction to the assumption that C_H is an s - t -edge-cut. \square

Lemma 1 implies that the optimal s - t -edge-cut in H has at most as many edges as the optimal valid s - t -edge-cut in G . Conversely, every cut in H gives a valid cut in G of at most the same cardinality: consider the sets C_G and C_H in the Algorithm `EDGECUT`, clearly $|C_G| \leq |C_H|$ holds. Thus, an optimal s - t -edge-cut in the two-layer model H yields an optimal valid s - t -edge-cut in the ToR graph G . The former can be found in polynomial time by network flow techniques [1]. This concludes the proof of Theorem 3. \square

3.4 Max Vertex-/Edge-Disjoint Valid s - t -Paths

Theorem 4. *For a given ToR graph $G = (V, E)$ and $s, t \in V$, finding the maximum number of vertex- respectively edge-disjoint valid s - t paths is NP-hard. Moreover, the number of paths is even inapproximable within a factor $2 - \varepsilon$ for any $\varepsilon > 0$, unless P equals NP.*

Proof. We will reduce the problem of finding two disjoint paths between two pairs of terminals in a directed graph to this problem. Let G be a directed graph and s_1, t_1, s_2, t_2 four distinct vertices of G . Form a ToR graph G' from G by adding two vertices s and t , and edges from s to s_1 , from t_1 to t , from t to s_2 and from t_2 to s . Note that no path s, t_2, \dots, t_1, t is valid. Thus, revealing the maximum number of vertex- respectively edge-disjoint valid s - t paths in G' would give a solution to the problem of finding two vertex- respectively edge-disjoint directed paths between s_1, t_1 and s_2, t_2 in G . The latter two problems are known to be NP-complete in general directed graphs [9].

This also directly gives the inapproximability gap. For an arbitrary $k \in \mathbb{N}$, we simply make k copies of the graph G' . Next, we identify all copies of s to one node and all copies of t to one node. Intuitively, we now have k “parallel” copies of G . Depending on G there are either k or $2k$ vertex- respectively edge-disjoint valid paths between s and t . Let $\varepsilon > 0$ be some constant, independent of k . Clearly, if a $(2 - \varepsilon)$ -approximation existed for max vertex- respectively edge-disjoint valid s - t paths, we could again solve the problem of finding two vertex- respectively edge-disjoint paths between s_1, t_1 and s_2, t_2 in G in polynomial time. \square

A Tight Approximation Algorithm For ease of presentation we focus on the max vertex-disjoint valid s - t paths problem and comment at the end of the section how the result can be transferred to the max edge-disjoint case. In order to state the approximation algorithm, we need some definitions. If a forward part of a valid s - t path p_1 intersects the backward part of a path p_2 at a node v , we call this a *crossing* at v . The two paths can be *recombined* at the crossing to form a new path, consisting of the first part of p_1 : s, \dots, v and the last part of p_2 : v, \dots, t . If p_1 and p_2 are recombined at v , the potential crossings on p_1 after node v and on p_2 before node v can be discarded. In the algorithm a recombination of such paths p_1 and p_2 may be revoked later on and thus not all these potential

crossings can be discarded. However, it will be possible to discard the crossings of one of the paths. Note that p_1 and p_2 can be the same; in particular, if a path p contains a forward and a backward part, which meet at node u , we also say that the two parts cross at u .

ALGORITHM VERTEXDISJOINTPATHS

1. From G construct the two-layer model H , compute max vertex-disjoint s - t paths \mathcal{P}_H in H .
 2. Interpret \mathcal{P}_H as set \mathcal{P}_G of valid s - t paths in G . Note: \mathcal{P}_G is not necessarily vertex-disjoint! Let \mathcal{F} denote the forward parts of paths in \mathcal{P}_G and \mathcal{B} the backward parts. Recombine the parts as follows:
 - (a) Select any not yet recombined forward part p_f in \mathcal{F} that has at least one *remaining* (i.e. not discarded, see below) crossing.
 - (b) Choose the first *remaining* crossing on p_f , let p_b in \mathcal{B} be the corresponding backward part.
 - (c) Recombine p_f and p_b , discard all previous crossings on p_b . In particular, if p_b was already recombined with p'_f , mark p'_f as not yet combined.
 - (d) Repeat until each forward part is either recombined or has no remaining crossings.
-

Theorem 5. *The algorithm VERTEXDISJOINTPATHS is a 2-approximation algorithm for the max vertex-disjoint valid s - t paths problem.*

Proof. We first prove that the algorithm actually outputs a set of vertex-disjoint paths, then mention why the running time is polynomial and finally show that the approximation ratio of 2 is achieved.

Note that since the paths \mathcal{P}_G are derived from vertex-disjoint s - t -paths in the two-layer model H , all forward parts \mathcal{F} are disjoint and also all backward parts \mathcal{B} . Let \mathcal{R}_i denote the set of recombined paths after the i th recombination. \mathcal{R}_0 is the empty set and thus vertex-disjoint. We now argue that if \mathcal{R}_i is vertex-disjoint, then also \mathcal{R}_{i+1} will be. In the $i+1$ st recombination the selected forward part p_f does not intersect any backward part in \mathcal{R}_i up to the chosen crossing, say at node v . Since step 2.(b) chooses the first remaining crossing on p_f , all potential crossings before v on p_f were discarded previously. We argue that also the rest of the backward part $p_b: v, \dots, t$ does not intersect any other path q in \mathcal{R}_i . Assume the contrary, then there is a path q in \mathcal{R}_i whose forward part q_f intersects p_b , say at node u . Let q_b be the backward part of q . Since the paths were derived from the two-layer model, at most two paths cross in each node. Thus q_f and q_b were recombined at a node $w \neq u$ and clearly w is after u on the forward part q_f (otherwise p_b would not intersect q_f). This gives the contradiction, since the algorithm would then have recombined q_f and p_b at node u instead of q_f and q_b at node w .

We have shown that the first part of p_f from s to v and the last part of p_b from v to t do not intersect any other path in \mathcal{R}_i . In case p_b was already

recombined in \mathcal{R}_i with some other forward part, this previous recombination is removed from \mathcal{R}_i , see step 2.(c). Thus, \mathcal{R}_{i+1} is vertex-disjoint. As each crossing is considered at most once for recombination, the number of recombinations is $O(|V|)$ and the running time is polynomial.

It remains to prove the approximation ratio. Assume that k is the optimal number of paths for a given instance. Clearly $|\mathcal{P}_G|$ is at least k . For each path p in \mathcal{P}_G either its forward part p_f , its backward part p_b , or both are recombined by the algorithm. If neither p_f nor p_b are recombined, p_f has at least one remaining crossing, namely the crossing with p_b . This crossing could not have been discarded, since p_b was never recombined with any forward part. Hence, either forward or backward part of each path are recombined, and thus at least $|\mathcal{P}_G|/2 \geq k/2$ disjoint valid s - t paths are found. \square

The algorithm can be easily adapted to the edge-disjoint paths setting. Here the crossings are at edges instead of nodes. The recombination of two paths that cross at an edge $e = (u, v)$ is done at node u , where e is directed from u to v . As in the computation of valid s - t -edge-cuts, $|V|$ parallel edges must be used to connect each vertex in the lower layer to its copy in the upper layer of the two-layer graph H ; this ensures that any number of paths can switch from the lower layer to the upper layer at the same node. Analogously to the proof of Theorem 5 we obtain:

Theorem 6. *There is a 2-approximation algorithm for the max edge-disjoint valid s - t paths problem.*

3.5 On the Gap between Disjoint Paths and Minimum Cuts

In the standard model of paths in directed or undirected graphs, Menger's theorem states that the maximum number of edge-disjoint s - t -paths is equal to the size of a minimum s - t -edge-cut, and the analogous result holds for vertex-disjoint paths and vertex-cuts (provided that there is no direct edge from s to t). As Menger's theorem applies to standard directed paths in the two-layer graph H , the proofs of Theorems 2 and 5 imply that for the valley-free path model there is always a valid s - t -vertex cut that is at most twice as large as the maximum number of vertex-disjoint valid s - t -paths. The same bound can be derived for the edge versions of the problems. Examples showing that the bound of 2 is tight are given in Fig. 3 both for the vertex version (left) and the edge version (right). In these examples, the size of a minimum valid s - t -cut is 2, while the maximum number of disjoint valid s - t -paths is 1.

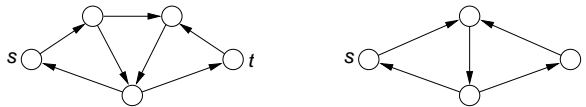


Fig. 3. ToR graphs demonstrating a gap of 2 between disjoint paths and cuts

4 Max Vertex-/Edge-Disjoint Valid s - t -Paths in DAGs

We consider the problem of computing vertex- or edge-disjoint paths in directed acyclic graphs. This is motivated by the consideration that in a strictly hierarchical network, one would obtain ToR graphs that are acyclic. Due to space limitations, we refer the reader to [7] for proofs of the following results. First, we obtain that the problems remain *NP*-hard even for acyclic graphs.

Theorem 7. *For a given acyclic ToR graph $G = (V, E)$ and $s, t \in V$, finding the maximum number of vertex- respectively edge-disjoint valid s - t -paths is *NP*-hard.*

In general ToR graphs, it is *NP*-hard to decide whether there are two edge- or vertex-disjoint valid paths from s to t (Theorem 4). For acyclic graphs, we are able to show that this decision problem can be solved in polynomial time for any constant number of paths. Our proof is based on an extension of a pebbling game introduced by Fortune et al. [9].

Theorem 8. *For a given acyclic ToR graph $G = (V, E)$, $s, t \in V$, and any constant k , one can decide in polynomial time if there exist k vertex-disjoint (edge-disjoint) valid paths between s and t in G (and compute such paths if the answer is yes).*

5 Conclusions

We have initiated the study of disjoint valid s - t -paths and valid s - t -cuts in the valley-free path model. These problems arise in the analysis of the AS topology of the Internet if commonly used routing policies are taken into account. The size of a minimum valid s - t -vertex-cut can be viewed as a reasonable measure of the robustness of the Internet connection between ASs s and t . If the minimum cut has size k , this means that k ASs must fail in order to completely disconnect s and t . Therefore, our algorithms could be useful for network administrators who want to assess the quality of their network's connection to the Internet. Note that our approximation algorithm for the min valid s - t -vertex-cut problem can be easily adapted to the weighted version of the problem (where an AS that is unlikely to fail can be given a large weight).

The problems we have studied may be seen as instances of a more general family of problems whose common theme is that the allowed paths in the graph must obey certain restrictions. One example of such a restriction are oriented paths (paths containing at least one directed edge) in mixed graphs (graphs with undirected and directed edges), as considered by Wanke [18] in the context of the analysis of different parcellation schemes of the macaque brain. Another example are paths in graphs with labeled edges where a path is allowed only if the sequence of its edge labels forms a word from a given formal language; shortest-path problems for this type of restriction are studied by Barrett et al. [4] in the context of transportation problems. It would be interesting to study the max disjoint s - t -paths problem and min s - t -cut problem in such a setting.

There are also several open problems for the valley-free path model. It would be useful to study whether the maximum edge-disjoint or vertex-disjoint valid s - t -paths problem can be approximated better for acyclic graphs, and it would be interesting to determine the complexity and approximability of the min valid s - t -vertex-cut problem for acyclic graphs.

References

1. A. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, N.J., 1993.
2. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties*. Springer, Berlin, 1999.
3. P. Baake and T. Wichmann. On the economics of Internet peering. *Netnomics*, 1(1), 1999.
4. C. L. Barrett, R. Jacob, and M. Marathe. Formal language constrained path problems. *SIAM J. Comput.*, 30(3):809–837, 2000.
5. E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis. The complexity of multiway cuts. *SIAM J. Comput.*, 4(23):864–894, 1994.
6. G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *Proceedings of INFOCOM'03*, 2003.
7. T. Erlebach, A. Hall, A. Panconesi, and D. Vukadinović. Cuts and disjoint paths in the valley-free path model. TIK-Report 180, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, 2003. Available electronically at <ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report180.pdf>.
8. T. Erlebach, A. Hall, and T. Schank. Classifying customer-provider relationships in the Internet. In *Proceedings of the IASTED International Conference on Communications and Computer Networks*, pages 538–545, 2002.
9. S. Fortune, J. Hopcroft, and J. Willie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.
10. L. Gao. On inferring Autonomous System relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, 2001.
11. N. Garg, V. Vazirani, and M. Yannakakis. Multiway cuts in directed and node weighted graphs. In *Proceedings of ICALP'94*, LNCS 820, pages 487–498, 1994.
12. G. Huston. Interconnection, peering and settlements—Part I. *Internet Protocol Journal*, March 1999.
13. G. Huston. Interconnection, peering and settlements—Part II. *Internet Protocol Journal*, June 1999.
14. C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkatachary. The impact of Internet policy and topology on delayed routing convergence. In *Proceedings of INFOCOM'01*, 2001.
15. L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proceedings of INFOCOM'02*, 2002.
16. H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proceedings of SPIE ITCOM'01*, 2001.
17. H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The impact of routing policy on Internet paths. In *Proceedings of INFOCOM'01*, 2001.
18. E. Wanke. The complexity of finding oriented paths in mixed graphs. Technical report, Institut für Informatik, Heinrich-Heine-Universität, Düsseldorf, 2003.