Supplementary Material for "PATHATTACK: Attacking Shortest Paths in Complex Networks"

Benjamin A. Miller[1], Zohair Shafi[1], Wheeler Ruml[2], Yevgeniy Vorobeychik[3], Tina Eliassi-Rad[1], and Scott Alfeld[4]

[1] Northeastern University, Boston MA 02115, USA
{miller.be, shafi.z, t.eliassirad}@northeastern.edu
[2] University of New Hampshire, Durham NH 03824, USA
ruml@cs.unh.edu
[3] Washington University in St. Louis, St. Louis MO 63130, USA
yvorobeychik@wustl.edu
[4] Amherst College, Amherst MA 01002, USA
salfeld@amherst.edu

A Notation

Table 1 lists the notation used in the paper.

B Problem Complexity

If each edge e has a cost of removal $c(e)$, the goal of Force Path Cut is to keep the total cost of edge removal within a given budget while forcing the shortest path to be p^*. (Cost of removal is not necessarily equal to the weight.) We can show that the 3-Terminal Cut problem reduces to this one. In k-Terminal Cut—described in [2]—we are given a weighted graph $G = (V, E)$, positive edge weights $w(e)$, and a budget b. There are k nodes from V designated as terminals. The goal is to find an edge subset $E' \subset E$, where $\sum_{e \in E'} w(e) \leq b$, whose removal disconnects all terminals from one another. This is shown in [2] to be NP-complete for $k > 2$. We will show that the solution to Force Path Cut would solve 3-Terminal Cut, thus proving the following theorem.

Theorem 1. *Force Path Cut is NP-complete for undirected graphs.*

To solve 3-Terminal Cut, we are given the graph $G = (V, E)$, with weights $w(e) > 0$ for all $e \in E$, and a budget $b > 0$, along with three terminal nodes $s_1, s_2, s_3 \in V$ (following notation from [2]). We will create a new graph $\hat{G}$ from G and use it as an input for Force Path Cut. Let w_{all} be the sum of all edge weights, i.e., $w_{\text{all}} = \sum_{e \in E} w(e)$. Note that if there are any edges between s_1, s_2, and s_3, these edges must be removed in the solution (any solution that does not remove them does not isolate the terminals from one another). Let E'_t be the set of any such edges, i.e.,

$$E'_t = E \cap \{\{s_1, s_2\}, \{s_2, s_3\}, \{s_1, s_3\}\}\,. \tag{1}$$

Symbol	Meaning
G	graph
V	vertex set
E	edge set
N	number of vertices
M	number of edges
s	source vertex
t	destination vertex
p^*	adversary's desired path in G
$w(e)$	edge weight function $w : E \to \mathbb{R}_{\geq 0}$
$\mathbb{R}_{\geq 0}$	set of nonnegative real numbers
$\mathcal{U}$	universe for the Set Cover problem
$\mathcal{S}$	set of subsets of universe $\mathcal{U}$
δ_S	binary vector representing inclusion of set $S \in \mathcal{S}$
$\mathbf{w}$	edge weight vector
$c(e)$	edge removal cost function $c : E \to \mathbb{R}_{\geq 0}$
$\mathbf{c}$	vector of edge removal costs
b	adversary's budget
Δ	binary vector representing edges cut
$\mathbf{x}$	binary vector representing edges in a path
P_{p^*}	set of paths from s to t no longer than p^*
$(\cdot)^\top$	matrix or vector transpose
$\langle k \rangle$	average degree in G
σ_k	standard deviation of node degrees in G
κ	global clustering coefficient in G
τ	transitivity in G
$\triangle$	number of triangles in G
φ	number of connected components in G

Table 1. Notation used throughout the paper

Since any edges between the terminals must be removed, the budget for removing the remaining edges must be reduced, yielding a new budget

$$\hat{b} = b - \sum_{e \in E'_t} w(e). \tag{2}$$

Finally, we add edges between the terminal nodes with specific weights: an edge between s_1 and s_2 and one between s_2 and s_3, each with weight $w_{\text{all}} + 2\epsilon$ for some $\epsilon > 0$, and an edge between s_1 and s_3 with weight $2w_{\text{all}} + 3\epsilon$. Let $\hat{E}$ be the edge set with the new edges added and $\hat{w}$ be the new set of edge weights (with all other weights retained from the original graph). The new graph $\hat{G} = \left(V, \hat{E}\right)$ will be an input to Force Path Cut, with budget $\hat{b}$, starting node $s = s_1$, destination node $t = s_3$, and target edge $p^* = (s_1, s_3)$. Pseudocode for this procedure is provided in Algorithm 1.

Algorithm 1 Create Force Path Cut input graph

Input: Graph $G = (V, E)$, weights w, budget $b \geq 0$, terminals $s_1, s_2, s_3 \in V$
Output: Graph $\hat{G}$, weights $\hat{w}$, unused edges E'_t
$w_{\text{all}} \leftarrow \sum_{e \in E} w(e)$
$E'_t \leftarrow E \cap \{\{s_1, s_2\}, \{s_2, s_3\}, \{s_1, s_3\}\}$
$\hat{E} \leftarrow E \setminus E'_t$
for all $e \in \hat{E}$ **do**
  $\hat{w}(e) \leftarrow w(e)$
end for
$e_{12} \leftarrow \{s_1, s_2\}$ $\quad \langle\langle$create new edges$\rangle\rangle$
$\hat{w}(e_{12}) \leftarrow w_{\text{all}} + 2\epsilon$
$e_{23} \leftarrow \{s_2, s_3\}$
$\hat{w}(e_{12}) \leftarrow w_{\text{all}} + 2\epsilon$
$e_{13} \leftarrow \{s_1, s_3\}$
$\hat{w}(e_{12}) \leftarrow 2w_{\text{all}} + 3\epsilon$
$\hat{E} \leftarrow \hat{E} \cup \{e_{12}, e_{23}, e_{13}\}$
return $\hat{G} = \left(V, \hat{E}\right), \hat{w}, E'_t$

Lemma 2. *Let $G = (V, E)$ be an undirected graph. For any node subset $V_s \subset V$, if E can be partitioned $E = E_s \cup E_{\bar{s}}$, $E_s \cap E_{\bar{s}} = \emptyset$, such that (1) all edges between nodes in V_s are in E_s and (2) there is no path between any two nodes in V_s within $E_{\bar{s}}$, then all simple paths between nodes in V_s use only edges in E_s.*

Proof. Suppose a simple path existed between two nodes $u, v \in V_s$ that included edges in $E_{\bar{s}}$. Let d be the number of edges in this path and let $e_i \in E$ for $1 \leq i \leq d$ be the sequence of edges starting from u and ending at v. Let j be the sequential index of the first edge in $E_{\bar{s}}$ that appears in the path. This edge goes from a vertex in V_s to a vertex in $V \setminus V_s$. (Any edges occurring beforehand are in E_s, so only connect nodes within V_s, and e_j is in $E_{\bar{s}}$, so at least one vertex is outside of V_s.) Let $e_j = \{u_j, v_j\}$, where $u_j \in V_s$ and $v_j \in V \setminus V_s$. Finally, let e_k be the first edge after e_j that connects a node from $V \setminus V_s$ to a node from V_s, i.e., the minimum $k > j$ such that $e_k = \{u_k, v_k\}$ where either $u_k \in V_s$ or $v_k \in V_s$. We note that such an edge must exist, since the final node in the sequence is in V_s. Note also that $e_k \in E_{\bar{s}}$, since one of its vertices is in $V \setminus V_s$. Without loss of generality, let $u_k \in V \setminus V_s$ and $v_k \in V_s$. The edges in the sequence e_i for $j \leq i \leq k$ form a path from $u_j \in V_s$ to $v_k \in V_s$ using only edges in $E_{\bar{s}}$. By the assumption of the lemma there is no path between any two nodes in V_s using edges in $E_{\bar{s}}$, this means that u_j and v_k must be the same node, which contradicts the premise that the path is simple. This completes the proof. □

Lemma 3. *Existence of a solution to 3-Terminal Cut implies existence of a solution to Force Path Cut in the graph modified by Algorithm 1.*

Proof. Let $E' \subset E$ be a solution to 3-Terminal Cut, i.e., a set of edges such that $\sum_{e \in E'} w(e) \leq b$ and in the graph $G' = (V, E \setminus E')$ there is no path connecting

Algorithm 2 Solve 3-Terminal Cut via Force Path Cut

Input: Graph $G = (V, E)$, weights w, budget $b \geq 0$, terminals $s_1, s_2, s_3 \in V$
Output: Boolean value indicating whether the 3 terminals can be separated
$\hat{G} = \left(V, \hat{E}\right), \hat{w}, E'_t \leftarrow$ output of Algorithm 1
$\hat{b} \leftarrow b - \sum_{e \in E'_t} w(e)$
if $\hat{b} < 0$ **then**
  return false $\langle\langle$budget is too small$\rangle\rangle$
end if
$s \leftarrow s_1$
$t \leftarrow s_3$
$p^* \leftarrow (s_1, s_3)$
return ForcePathCut$\left(\hat{G}, \hat{w}, \hat{w}, \hat{b}, p^*, s, t\right)$

any of the terminal nodes s_1, s_2, and s_3. Any edges from G that directly connect the terminals must be in E' or such a path would exist. Letting E'_t be the set of any such edges (as in (1)), this means that $E'_t \subset E'$.

The graph as modified by Algorithm 1 includes edges

$$\hat{E} = (E \setminus E'_t) \cup \{e_{12}, e_{13}, e_{23}\},$$

with weights $w(e_{12}) = w_{\text{all}} + 2\epsilon$, $w(e_{23}) = w_{\text{all}} + 2\epsilon$, and $w(e_{13}) = 2w_{\text{all}} + 3\epsilon$. By the assumption of the lemma, removing all edges in E' disconnects the terminals from one another. Algorithm 1 starts by removing E'_t and adds 3 new edges to the graph, resulting in the graph $\hat{G}$. Consider a partition of $\hat{E}$—the edges in $\hat{G}$—into 3 subsets:

$$E_1 = \{e_{12}, e_{23}, e_{13}\} \tag{3}$$
$$E_2 = (E' \setminus E'_t) \tag{4}$$
$$E_3 = E \setminus E'. \tag{5}$$

Note that $\hat{E} = E_1 \cup E_2 \cup E_3$ and

$$E_1 \cap E_2 = E_2 \cap E_3 = E_1 \cap E_3 = \emptyset, \tag{6}$$

so (3)–(5) describe a proper partition. Suppose the Force Path Cut procedure removes the edges in E_2 from $\hat{G}$. Note that

$$\sum_{e \in E'} w(e) \leq b \Rightarrow \sum_{e_1 \in E'_t} w(e) + \sum_{e_2 \in E' \setminus E'_t} w(e) \leq b \tag{7}$$
$$\Rightarrow \sum_{e_2 \in E' \setminus E'_t} w(e) \leq b - \sum_{e_1 \in E'_t} w(e) \tag{8}$$
$$\Rightarrow \sum_{e_2 \in E_2} \hat{w}(e) \leq \hat{b}, \tag{9}$$

with $\hat{b}$ as defined in Algorithm 2. This implies that the edges in E_2 would be within the budget allocated to Force Path Cut.

After removing the edges in E_2, the remaining edges in $\hat{G}$ would be $E_1 \cup E_3$. The set E_1 only includes edges among terminals and, by the assumption of the lemma, E_3 does not include any path between any two terminal nodes. Thus, by Lemma 2, any path from s_1 to s_3 after removing E_2 includes only edges in E_1.

There are therefore two possible paths from s_1 to s_3: $s_1 \to s_3$ and $s_1 \to s_2 \to s_3$. The latter path has weight

$$\hat{w}(e_{12}) + \hat{w}(e_{23}) = 2(w_{\text{all}} + 2\epsilon) > 2w_{\text{all}} + 3\epsilon = \hat{w}(e_{13}), \tag{10}$$

and thus the former is the shortest path from s_1 to s_3 in $\hat{G}$. Since $p^* = (s_1, s_3)$, the shortest path from $s = s_1$ to $t = s_3$ is p^*, meaning that if there is a solution to 3-Terminal Cut in G, there is a solution to Force Path Cut in $\hat{G}$. □

Lemma 4. *A solution to Force Path Cut in the graph modified by Algorithm 1 implies a solution to 3-Terminal Cut in the original graph.*

Proof. Given a graph $G = (V, E)$, weights w, a budget b, and terminals

$$s_1, s_2, s_3 \in V,$$

use Algorithm 1 to compute $\hat{G}$, $\hat{w}$, and E'_t. As in Algorithm 2, compute $\hat{b}$, set s, t, and p^* and solve Force Path Cut. Let $\hat{E}'$ be the edges that are cut when solving the problem, meaning (1) the shortest path from $s = s_1$ to $t = s_3$ is p^* and (2) $\sum_{e \in \hat{E}'} \hat{w}(e) \leq \hat{b}$.

After removing the edges, consider a partition of the remaining edge set into $\{e_{12}, e_{23}, e_{13}\}$ and its complement

$$\hat{E}_{\bar{t}} = (\hat{E} \setminus \hat{E}') \setminus \{e_{12}, e_{23}, e_{13}\}. \tag{11}$$

Within $\hat{E}_{\bar{t}}$, there is no path between any two terminal nodes. If there were any such path, it would have length at most w_{all}, since

$$\sum_{e \in \hat{E}_{\bar{t}}} \hat{w}(e) \leq w_{\text{all}}. \tag{12}$$

Existence of such a path would have at least one of the following implications:

- If such a path p existed between s_1 and s_2, then p followed by e_{23} would be a path from s_1 to s_3 with length at most

$$w_{\text{all}} + \hat{w}(e_{23}) = 2w_{\text{all}} + 2\epsilon < \hat{w}(e_{13}). \tag{13}$$

  This implies that $s_1 \to s_3$ is not the shortest path from s_1 to s_3, thus contradicting the assumption of the lemma.
- The analogous case for a path from s_2 to s_3 yields an analogous contradiction.
- If such a path existed between s_1 and s_3, its length would be at most $w_{\text{all}} < \hat{w}(e_{13})$, again contradicting the assumption that $s_1 \to s_3$ is the shortest path from s_1 to s_3 in $\hat{G}'$.

Thus, if we remove $\hat{E}'$ from $\hat{G}$, the only edges connecting the terminal nodes are e_{12}, e_{23}, and e_{13}. In other words, removing $\hat{E}'$ and $\{e_{12}, e_{23}, e_{13}\}$ from $\hat{E}$ will result in the terminals being disconnected from one another. Note that E and $\hat{E}$ are the same after removing nodes between the terminals, i.e.,

$$E \setminus E'_t = \hat{E} \setminus \{e_1, e_2, e_3\} \tag{14}$$

This means that

$$\begin{aligned}(\hat{E} \setminus \hat{E}') \setminus \{e_1, e_2, e_3\} &= (\hat{E} \setminus \{e_1, e_2, e_3\}) \setminus \hat{E}' && (15)\\ &= (E \setminus E'_t) \setminus \hat{E}' && (16)\\ &= E \setminus (E'_t \cup \hat{E}'). && (17)\end{aligned}$$

Thus, removing E'_t and $\hat{E}'$ from the original graph results in a graph with no path between any two terminals. Recall that the assumption of the lemma requires:

$$\begin{aligned}&\sum_{e \in \hat{E}'} \hat{w}(e) \leq \hat{b} = b - \sum_{e \in E'_t} w(e) && (18)\\ \Rightarrow &\sum_{e_1 \in E'_t} w(e_1) + \sum_{e_2 \in \hat{E}'} \hat{w}(e_2) \leq b && (19)\\ \Rightarrow &\sum_{e \in E'_t \cup \hat{E}'} w(e) \leq b. && (20)\end{aligned}$$

Removing these edges is, therefore, within the budget allocated. □

We have now proven that Algorithm 2 is a polynomial-time reduction from 3-Terminal Cut to Force Path Cut, as Lemmas 3 and 4 have shown. Since 3-Terminal Cut is NP-complete, this implies Force Path Cut is NP-complete as well.

C Datasets

Our experiments were run on several synthetic and real networks across different edge-weight initialization. All networks are undirected. We described the edge-weight initialization schemes in Section 4.2 of the paper. Table 2 provides summary statistics of the synthetic networks.

We ran experiments on both weighted and unweighted real networks. In cases of unweighted networks, we added either Poisson, uniformly distributed, or equal weights as was the case of synthetic networks. Below is a brief description of each network used. Table 3 summarizes the properties of each network.

The unweighted networks are:

- Wikispeedia graph (WIKI): The network consists of Web pages (nodes) and connections (edges) created from the user-generated paths in the Wikispeedia game [6]. Available at `https://stanford.io/3cLKDb7`.

Networks	Nodes	Edges	$\langle k \rangle$	σ_k	κ	τ	$\triangle$	φ
ER	16,000 ±0	159,880 ±38	19.985 ±0.05	4.469 ±0.02	0.001 ±0.0	0.001 ±0.0	1,326 ±39	1 ±0
BA	16,000 ±0	159,900 ±0	19.987 ±0	24.475 ±0.3	0.007 ±0.0	0.006 ±0	17,133 ±500	1 ±0
KR	16,337 ±22	159,595 ±94	19.537 ±0.02	16.537 ±1.32	0.003 ±0	0.005 ±0.002	8,492 ±2,234	1.18 ±0.38
LAT	81,225 ±0	161,880 ±0	3.985 ±0	0.118 ±0	0 ±0	0 ±0	0 ±0	1 ±0
COMP	565 ±0	159,330 ±0	564 ±0	0 ±0	1 ±0	1 ±0	29,900,930 ±0	1 ±0

Table 2. Properties of the synthetic networks used in our experiments. For each random graph model, we generate 100 networks. Note that the number of edges across the different networks is $\approx$ 160K. The table shows the average degree ($\langle k \rangle$), standard deviation of the degree (σ_k), global clustering coefficient (κ), transitivity (τ), number of triangles ($\triangle$), and the number of components (φ). The $\pm$ values show the standard deviation across 100 runs of each random graph model.

- Oregon autonomous system network (AS): Nodes represent autonomous systems of routers and edges denote communication between the systems [4]. The dataset was collected at the University of Oregon on 31 March 2001. Available at `https://stanford.io/3rLItfN`.
- Pennsylvania road network (PA-ROAD): Nodes are intersections in Pennsylvania, connected by edges representing roads [5]. Available at `https://stanford.io/31Jnb7W`.

Networks	Nodes	Edges	$\langle k \rangle$	σ_k	κ	τ	$\triangle$	φ
GRID	347	444	2.559	1.967	0.086	0.087	40	1
LBL	3186	9486	5.954	25.515	0.099	0.005	1821	10
WIKI	4,592	106,647	46.449	69.878	0.274	0.102	550,545	2
AS	10,670	22,002	4.124	31.986	0.296	0.009	17,144	1
PA-ROAD	1,088,092	1,541,898	2.834	1.016	0.046	0.059	67,150	206
NEUS	1,524,453	1,934,010	2.537	0.950	0.022	0.030	37,012	1
DBLP	1,836,596	8,309,938	9.049	21.381	0.631	0.165	26,912,200	60,512

Table 3. Properties of the real networks used in our experiments. For each network, we are listing the average degree ($\langle k \rangle$), standard deviation of the degree (σ_k), global clustering coefficient (κ), transitivity (τ), number of triangles ($\triangle$), and the number of components (φ).

The weighted networks are:

- Central Chilean Power Grid (GRID): Nodes represent power plants, substations, taps, and junctions in the Chilean power grid. Edges represent transmission lines, with distances in kilometers [3]. The capacity of each line in kilovolts is also provided. Available at `https://bit.ly/2OjqCPO`.

- Lawrence Berkeley National Laboratory network data (LBL): A graph of computer network traffic, which includes counts of the number of connections between machines over time. Counts are inverted for use as distances. Available at `https://bit.ly/2PQbOsr`.
- Northeast US Road Network (NEUS): Nodes are intersections in the northeastern part of the United States, interconnected by roads (edges), with weights corresponding to distance in kilometers. Available at `https://bit.ly/2QWcug9`.
- DBLP coauthorship graph (DBLP): This is a co-authorship network [1]. We invert the number of co-authored papers to create a distance (rather than similarity) between the associated authors. Available at `https://bit.ly/3fytXFS`.

D Number of Constraints

In Figure 1, we illustrate the number of constraints used by `PATHATTACK` and the baseline algorithms. In over 99% of experiments, the number of constraints used by `PATHATTACK` is less than 5% of the number of edges, and in the largest graphs (PA-ROAD, NEUS, DBLP) it is tens of thousands of times smaller.

References

1. Benson, A.R., Abebe, R., Schaub, M.T., Jadbabaie, A., Kleinberg, J.: Simplicial closure and higher-order link prediction. Proc. Nat. Acad. Sci. **115**(48), E11221–E11230 (2018)
2. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. SIAM J. Comput. **23**(4), 864–894 (1994)
3. Kim, H., Olave-Rojas, D., Álvarez-Miranda, E., Son, S.W.: In-depth data on the network structure and hourly activity of the central Chilean power grid. Sci. Data **5**(1), 1–10 (2018)
4. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: Densification laws, shrinking diameters and possible explanations. In: KDD. pp. 177–187 (2005)
5. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics **6**(1), 29–123 (2009)
6. West, R., Pineau, J., Precup, D.: Wikispeedia: An online game for inferring semantic distances between concepts. In: IJCAI (2009)

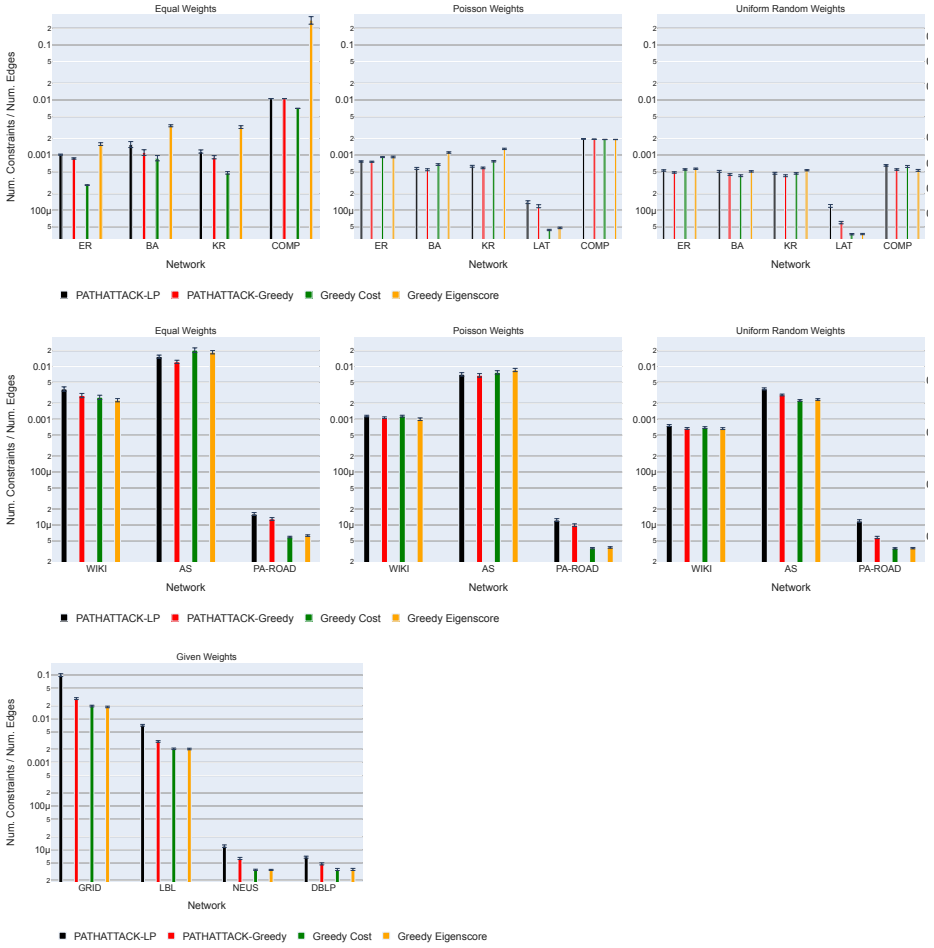

Fig. 1. Total number of constraints used to approximate the optimal solution to Force Path Cut. Results are shown for the two variants of `PATHATTACK` and the two baselines. Bar height indicates the number of paths explicitly considered as a proportion of the number of edges in the graph. Error bars denote standard errors. Results are shown for synthetic networks (top row), real networks with randomly generated weights (middle row), and real weighted networks (bottom row). In all cases, the number of constraints used by `PATHATTACK` is less than 10% of the number of edges, and in most cases is several orders of magnitude smaller. A small number of constraints has positive implications for both the running time and the approximation bound.