# Linear Loss Function for the Network Blocking Game: An Efficient Model for Measuring Network Robustness and Link Criticality

Aron Laszka[1], Dávid Szeszlér[2], and Levente Buttyán[1]

[1] Laboratory of Cryptography and System Security,
Department of Telecommunications,
Budapest University of Technology and Economics
{laszka, buttyan}@crysys.hu
[2] Department of Computer Science,
Budapest University of Technology and Economics
szeszler@cs.bme.hu

**Abstract.** In order to design robust networks, first, one has to be able to measure robustness of network topologies. In [1], a game-theoretic model, the network blocking game, was proposed for this purpose, where a network operator and an attacker interact in a zero-sum game played on a network topology, and the value of the equilibrium payoff in this game is interpreted as a measure of robustness of that topology. The payoff for a given pair of pure strategies is based on a loss-in-value function. Besides measuring the robustness of network topologies, the model can be also used to identify critical edges that are likely to be attacked. Unfortunately, previously proposed loss-in-value functions are either too simplistic or lead to a game whose equilibrium is not known to be computable in polynomial time. In this paper, we propose a new, linear loss-in-value function, which is meaningful and leads to a game whose equilibrium is efficiently computable. Furthermore, we show that the resulting game-theoretic robustness metric is related to the Cheeger constant of the topology graph, which is a well-known metric in graph theory.

**Keywords:** game theory, adversarial games, network robustness, computational complexity, blocking games, Cheeger constant

## 1 Introduction

In order to be able to design networks that resist malicious attacks and accidental failures, one must be able, first of all, to measure the robustness of network topologies. A number of graph-theoretic robustness metrics, such as node and edge connectivity, graph strength [2], toughness [3], and persistence [4], can be used for this purpose. Recently, however, another approach for measuring the robustness of network topologies has been proposed by Gueye *et al.* in a series of papers [5,6,1]. In their approach, the robustness of a network topology is

characterized by the equilibrium payoff in a two-player zero-sum game played by a network operator and an attacker.

In this model, the network operator chooses a spanning tree of the topology graph to be used for routing messages in the network, and simultaneously, the attacker chooses an edge of the topology graph to be removed. If the edge chosen by the attacker happens to be in the spanning tree chosen by the operator, then the attacker's payoff is positive and the operator's payoff is negative, otherwise they both receive 0 as payoff. In the former case, the actual value of the payoff depends on the loss-in-value function that is used for characterizing the effect of an edge being removed from the spanning tree. For instance, in [5], a simple indicator function is used: if the edge removed by the attacker is in the spanning tree, then the attacker's payoff is 1, otherwise it is 0. Another example is given in [7], where the payoff for the attacker is equal to the number of nodes that the attacker separates from a designated node in the spanning tree (e.g., a gateway in an access network) by removing the chosen edge.

The optimal strategies of such an attacker-defender game can be used to *identify critical edges* that are likely to be attacked, and the equilibrium payoff can be interpreted as a *measure of robustness* of the network topology at hand. In some cases, the game-theoretic robustness of a network may even be directly related to a graph-theoretic robustness metric. For instance, in [7], it is shown that the equilibrium payoff of the game where the loss-in-value function is defined as the number of nodes that the attacker separates from a designated node in the network is equal to the reciprocal of the persistence of the network as defined in [4]. Hence, the game-theoretic model can provide additional insights into the understanding of the graph-theoretic robustness metrics.

Unfortunately, the loss-in-value functions of [5] and [7] can not be used generally: The former is too simplistic as it does not take into account the magnitude of the damage caused by the attack. The latter is concerned with only those types of networks, where the nodes have to communicate only with a designated node, such as access and sensor networks. In [1], a number of loss-in-value functions, derived from previously proposed network value functions, are introduced and studied. However, to the best of our knowledge, finding efficient algorithms to compute the equilibrium payoff or the optimal strategies in case of these loss-in-value functions is still an open question.

Our contributions in this paper are the following: We propose a new, linear loss-in-value function, for which we also provide strongly polynomial-time algorithms to compute optimal adversarial and operator strategies and, thus, the payoff in the Nash equilibria of the game. This means that we can compute the game-theoretic robustness efficiently in case of this linear loss-in-value function. Moreover, our proposed linear loss-in-value function is meaningful in the sense that it is lower- and upper-bounded by loss-in-value functions previously proposed in [1]. In addition, we prove that the payoff in the Nash equilibria is closely related to the Cheeger constant of a graph (also called minimum edge expansion or isoperimetric number), a well-known metric in graph theory. Thus,

we can relate the game-theoretic robustness metric resulting from the proposed loss-in-value function to a graph-theoretic robustness notion.

The organization of this paper is the following: In Section 2, we briefly summarize previous related results. In Section 3, we describe the game model and introduce our linear loss-in-value function. In Section 4 and 5, we propose optimal operator and adversarial strategies and show how to compute them very efficiently. In Section 6, we combine the results of the previous sections to study the Nash equilibria of the game. In Section 7, we discuss properties of the optimal adversarial strategies. In Section 8, we show how the equilibrium payoff is related to the Cheeger constant. In Section 9, we generalize the game model to allow nodes with non-uniform weights. Finally, in Section 10, we conclude the paper.

## 2 Related Work

In [5], the strategic interactions between a network operator, whose goal is to keep a network connected, and an attacker, whose goal is to disconnect the network, were modeled as a two-player, one-shot, zero-sum game: The operator chooses a spanning tree $T$ of the network $G$ as the communication infrastructure while the adversary chooses a link $e$ as the target of her attack. The payoff of the adversary (or the loss of the operator) is $1_{e \in T}$, i.e., it is 1 if the targeted link is part of the chosen spanning tree, and 0 otherwise [3] . It was shown that the payoff in every Nash equilibrium of the game is equal to the reciprocal of the *(undirected) strength* of the network $\sigma(G)$, which can be computed efficiently. Furthermore, an efficient algorithm was provided to compute an optimal adversarial strategy.

In [6], the model was generalized to include link attack costs, which can vary based on the targeted links, resulting in a non-zero-sum game. Efficient algorithms were provided to compute the payoff in the Nash equilibria and to obtain an optimal adversarial strategy.

In [8], the model was generalized to incorporate link faults, which are random malfunctions independent of the adversary. In this model of interdependent reliability and security, the operator knows the distribution of link faults and the relative frequencies of faults and attacks, while the role of the adversary remains the same as in the basic model. Efficient algorithms were provided for two particular link fault distributions, the uniform distribution and a special distribution with a critical link being more vulnerable.

In [1], the indicator function $1_{e \in T}$ was replaced with a general *loss-in-value* function $\lambda(T, e)$, which quantifies the gain of the adversary and the loss of the operator when link $e$ is targeted and communication is carried over tree $T$. Previously proposed models for the value of a network were used to derive various loss-in-value functions. Some of the proposed loss-in-value functions (Metcalfe,

[3] The details of the model are described in Section 3. Here, we only introduce the basic concepts that are necessary to the discussion of related work.

Reed, BOT), as well as the indicator loss-in-value function $1_{e\in T}$ (termed GWA) are plotted in Figure 1.

In [7], the interactions in a many-to-one network, such as an access network or sensor network, were studied using a special loss-in-value function. The proposed function measures the number (or total value) of the nodes that are separated from a designated node when an attack occurs. It was shown that the payoff in every Nash equilibrium of the game is equal to the reciprocal of the *persistence* (or *directed strength*) of the network $\pi(G)$, which can be computed efficiently. Furthermore, efficient algorithms were provided to compute optimal operator and adversarial strategies.

## 3 Game Model

The network topology is represented by a connected undirected simple graph $G = (V, E)$. The goal of the network operator is to keep the nodes of the network connected to each other, while the goal of the adversary is to separate the nodes from each other.

The interaction between the network operator and the adversary is modeled as a two-player, one-shot, zero-sum game. The network operator chooses a spanning tree to be used for communications. The mixed strategy of the network operator is a distribution on the set of spanning trees $\mathcal{T}(G)$, i.e., $\mathcal{A} := \{\alpha \in \mathbb{R}_{\geq 0}^{|\mathcal{T}(G)|} | \sum_{T\in\mathcal{T}(G)} \alpha_T = 1\}$. The adversary chooses an edge to be attacked. The mixed strategy of the adversary is a distribution on the set of edges $E(G)$, i.e., $\mathcal{B} := \{\beta \in \mathbb{R}_{\geq 0}^{|E(G)|} | \sum_{e\in E(G)} \beta_e = 1\}$.

The payoff of the adversary (or the loss of the operator) is given by the loss-in-value function $\lambda(T, e)$. Thus, the expected payoff of the adversary is

$$\sum_{e\in E(G)} \sum_{T\in\mathcal{T}(G)} \alpha_T \beta_e \lambda(T, e) \ , \tag{1}$$

which the adversary tries to maximize and the operator tries to minimize.

### 3.1 Our Proposed Loss Function

In this paper, we propose a "linear" loss-in-value function, denoted by $\lambda(T, e)$, where $T$ and $e$ are the spanning tree and edge chosen by the operator and the adversary, respectively. If $e \in T$, then let $\lambda(T, e)$ be the number of nodes in the smaller component of $G[T \setminus \{e\}]$, where $G[F]$ denotes the graph $G' = (V(G), F)$, i.e., $\lambda(T, e)$ is the number of nodes that are separated from the larger connected component after the attack. If $e \notin T$, then let $\lambda(T, e) = 0$, i.e., there is no loss if the spanning tree remains intact. More formally:

**Definition 1 (Linear loss-in-value function).**

$$\lambda(T, e) := \begin{cases} \min_{C\in \text{ components of } G[T\setminus\{e\}]} |C|, & \text{if } e \in T \\ 0, & \text{if } e \notin T \ . \end{cases} \tag{2}$$

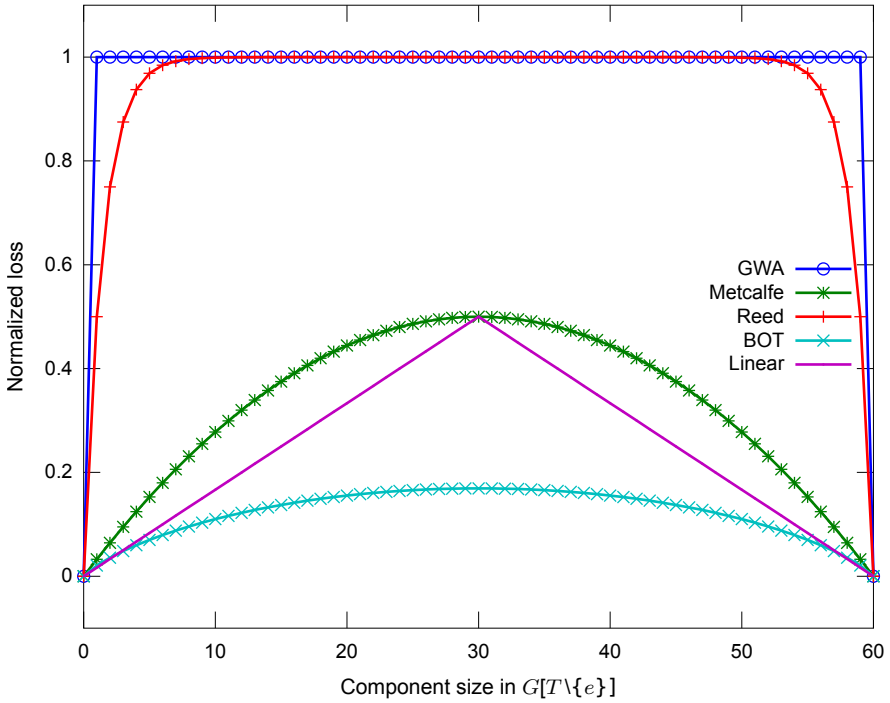



Fig. 1: Comparison of different loss-in-value functions.

Figure 1 compares our linear loss-in-value function to some of the functions proposed in [1]. The comparison is performed in a network consisting of 60 nodes. The horizontal axis shows the sizes of the components of the graph resulting from the attack. Extreme values 0 and 60 correspond to intact networks, i.e., when $e \notin T$. Values $0 < n < 60$ correspond to a damaged network consisting of two components of size $n$ and $60 - n$. The vertical axis measures the payoff of the adversary or, equivalently, the loss of the operator. The previously proposed loss functions are "normalized" in the manner described in [1]: each loss function is divided by the value of the intact network, which is determined by the corresponding network value function. Our loss function is not based on a network value function, but, since each node being separated causes a loss of value 1, we can establish that the value of the intact network is the number of nodes $|V(G)|$. Therefore, our linear loss function is "normalized" by dividing it with the constant coefficient $|V(G)|$. The normalization allows us to make an unbiased comparison between the different loss-in-value functions.

The figure shows that our linear loss function is bounded by the previously proposed loss functions Metcalfe and BOT. The Metcalfe function measures a special quadratic loss and is an upper bound of our function. The BOT function measures a special logarithmic loss and is a lower bound of our function. For

the exact definitions and a discussion of these functions we refer the reader to [1]. We can conclude that our linear loss function is at least as realistic as the previously proposed functions.

Please recall that, to the best of our knowledge, there are no polynomial-time algorithms known to compute the equilibria or the optimal strategies of the games based on the above loss functions, except for the GWA function. For further comparison between different loss functions, see Section 6.1.

## 4 Operator Strategy

In this section, we propose an operator strategy. Later, in Section 5, we show that this strategy is optimal by proving that it attains the lowest possible expected loss for the operator if the adversary is rational.

**Definition 2 (Expected loss).** *The* expected loss *(or* importance*) of an edge $e \in E(G)$ in a given operator strategy $\alpha$ is the expected payoff of the pure adversarial strategy targeting exclusively $e$, i.e., $\sum_{T \in \mathcal{T}} \alpha_T \cdot \lambda(T, e)$.*

To obtain an optimal operator strategy, consider the following linear program:

Variables:

$$\forall r \in V(G): \ \alpha_r \in \mathbb{R}_{\geq 0}$$
$$\forall r \in V(G)\ \forall \{u, v\} \in E(G): \ f_r(u, v), f_r(v, u) \in \mathbb{R}_{\geq 0}$$

Objective:

$$\text{maximize} \sum_{r \in V(G)} \alpha_r \tag{3}$$

Constraints:

$$\forall \{u, v\} \in E(G): \sum_{r \in V(G)} f_r(u, v) + f_r(v, u) \leq 1 \tag{4}$$

$$\forall r \in V(G)\ \forall v \in V(G) \setminus \{r\}: \sum_{\{u,v\} \in E(G)} f_r(v, u) - f_r(u, v) \geq \alpha_r \ . \tag{5}$$

Let $h'(G)$ denote the optimal value of the above linear program.

The above linear program can be viewed as a special multi-commodity flow problem: There is a commodity for every $r \in V(G)$. For every commodity, $r$ is a sink and every other node is a source producing $\alpha_r$. For the commodity consumed by $r$, the amount of flow from $u$ to $v$ is given by $f_r(u, v)$. Finally, every edge has a capacity of 1.

**Theorem 1.** *There is an operator strategy that achieves at most $\frac{1}{h'(G)}$ loss for the operator (regardless of the strategy of the adversary).*

*Proof.* Our goal is to find a distribution $\overline{\boldsymbol{\alpha}}$ such that

$$\forall e \in E: \ \sum_{T \in \mathcal{T}} \overline{\alpha}_T \lambda(T, e) \leq \frac{1}{h'(G)} \ , \tag{6}$$

i.e., the expected loss of every edge is at most $\frac{1}{h'(G)}$. Equivalently, we have to find weights $\boldsymbol{\alpha} \geq \mathbf{0}$ such that

$$\forall e \in E: \ \sum_{T \in \mathcal{T}} \alpha_T \lambda(T, e) \leq 1 \ , \tag{7}$$

i.e., the expected loss of every edge is at most one, and

$$\sum_{T \in \mathcal{T}} \alpha_T = h'(G) \ . \tag{8}$$

Our proof is constructive and it is based on the following algorithm:

1. Solve the above LP.
2. For each $r \in V(G)$,
   - find a set of weighted spanning trees $T_r^1, \ldots, T_r^{m_r}$ with a total weight of $\alpha_r$ that satisfies the constraint that the expected loss of each edge $\{u, v\}$ is less than or equal to $f_r(u, v) + f_r(v, u)$ using the flow decomposition algorithm proposed in [7]. The details of this algorithm can be found in Appendix A.

We claim that the resulting set of spanning trees $T_r^1, \ldots, T_r^{m_r}$ and corresponding coefficients $\alpha_r^1, \ldots, \alpha_r^{m_r}$ as a strategy satisfy that the expected loss of every edge is at most $\frac{1}{h'(G)}$.

Firstly, we have

$$\forall r \in V: \ \sum_{i=1}^{m_r} \alpha_r^i = \alpha_r \tag{9}$$

from the flow decomposition algorithm and

$$\sum_{r \in V} \alpha_r = h'(G) \tag{10}$$

by definition. Therefore, the total weight of the spanning trees is equal to $h'(G)$.

Let $\lambda_r(T, e)$ denote the number of nodes that are separated from $r \in V(G)$ in $G[T \setminus e]$, i.e., the number of nodes that are not in the same component as $r$ in $G[T \setminus e]$. Then, we have

$$\forall \{u, v\} \in E, r \in V: \sum_{i=1}^{m_r} \alpha_r^i \lambda_r(T_r^i, \{u, v\}) \leq f_r(u, v) + f_r(v, u) \tag{11}$$

from the flow decomposition algorithm.

By definition, $\lambda(T,e) \leq \lambda_r(T,e)$ as $\lambda(T,e)$ is the number of nodes in the not larger component (or 0, if there is only one component). Since $\lambda(T_r^i,e) \leq \lambda_r(T_r^i,e)$,

$$\forall\{u,v\} \in E : \sum_{r\in V} \sum_{i=1}^{m_r} \alpha_r^i \lambda(T_r^i,\{u,v\}) \leq \sum_{r\in V} f_r(u,v) + f_r(v,u) \ . \tag{12}$$

We also have

$$\forall\{u,v\} \in E : \ \sum_{r\in V} f_r(u,v) + f_r(v,u) \leq 1 \tag{13}$$

from the definition of the LP. Therefore, the expected loss of every edge (LHS of Equation 12)is at most 1. □

The following theorem shows how efficient the above algorithm is:

**Theorem 2.** *The operator strategy described in Theorem 1 can be computed in strongly polynomial time and its support, i.e., the set of spanning trees that have nonzero probability, consists of at most $|V(G)| \cdot |E(G)|$ spanning trees.*

*Proof.* We show that both steps of the algorithm presented in the proof of Theorem 1 can be performed in strongly polynomial time. In [9], a polynomial linear programming algorithm was presented whose number of arithmetic steps depends only on the size of the numbers in the constraint matrix. Since, in our case, the constraint matrix consists of only values of $-1$, 0 and 1, the linear programming problem can be solved in strongly polynomial time. It can be easily verified that the flow decomposition algorithm also runs in strongly polynomial time: Clearly, the number of arithmetic steps in every iteration (see Appendix A) of the algorithm is polynomial. In [7], it was shown that there are at most $|E(G)|$ iterations. Since the algorithm has to be run once for every node, the total number of iterations is at most $|V(G)| \cdot |E(G)|$.

In [7], it was also shown shown that the support of the distribution produced by the flow decomposition algorithm consists of at most $|E(G)|$ spanning trees. Since at most $|V(G)|$ flows have to be decomposed, the support of the resulting strategy has at most $|V(G)| \cdot |E(G)|$ spanning trees. □

The first part of the theorem states that our algorithm is very efficient as it is not only polynomial-time, but strongly polynomial-time, while the second part of the theorem states that resulting strategy is surprisingly simple.

## 5 Adversarial Strategy

In this section, we propose an optimal adversarial strategy, which attains $\frac{1}{h'(G)}$ expected payoff, regardless of the strategy of the operator. We have already shown in the previous section that this is the best attainable payoff for the adversary if the operator is rational.

**Lemma 1.** *For every spanning tree $T$, there exists a spanning reverse arborescence* [4] *such that*

- *the arborescence consists of the edges of $T$ and*
- *each arc $e \in T$ is directed such that its target is in the larger component of $G[T \setminus \{e\}]$* [5] *.*

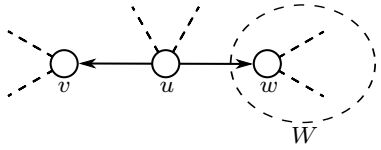



Fig. 2: Illustration for the proof of Lemma 1.

*Proof.* Direct each edge $e$ of $T$ such that its target is not in the smaller component of $G[T\setminus\{e\}]$. We have to prove that the result is indeed an arborescence, i.e., there is no pair of arcs $(u,v),(u,w) : u,v,w \in V(G), v \neq w$. Assume that this is not true: Let $W$ denote the node set of the not smaller component of $G[T \setminus \{(u,w)\}]$. Since $W$ consists of the nodes of the larger component, $|W \cup \{u\}| > \frac{|V(G)|}{2}$. But this leads to a contradiction as $W \cup \{u\}$ is a subset of the smaller component of $G[T \setminus \{(u,v)\}]$. See Figure 2 for an illustration. □

Now, consider the dual of the linear program introduced in the previous section:
Variables:

$$\forall e \in E(G): \ \beta_e \in \mathbb{R}_{\geq 0}$$
$$\forall r, v \in V(G), r \neq v: \ \pi_r(v) \in \mathbb{R}_{\geq 0}$$

Objective:

$$\text{minimize} \sum_{e \in E(G)} \beta_e \tag{14}$$

Constraints:

$$\forall r \in V(G): \sum_{v \in V(G)\setminus\{r\}} \pi_r(v) \geq 1 \tag{15}$$
$$\forall r \in V(G)\ \forall \{u,v\} \in E(G): \ \beta_{\{u,v\}} - \pi_r(u) + \pi_r(v) \geq 0 \tag{16}$$
$$\forall r \in V(G)\ \forall \{u,v\} \in E(G): \ \beta_{\{u,v\}} + \pi_r(u) - \pi_r(v) \geq 0 \ , \tag{17}$$

where $\pi_r(r) \equiv 0$ to simplify the equations.

[4] A directed, rooted spanning tree in which all edges point to the root.
[5] If the two components are of equal size, then the direction is arbitrary.

Variable $\pi$ can be viewed as a set of $|V(G)|$ potential functions, one for each node $r \in V(G)$. For every potential function $\pi_r$, the potential difference between two adjacent nodes connected by edge $e$ is bounded by the edge weight $\beta_e$.

The last two constraints can be written as

$$|\pi_r(u) - \pi_r(v)| \leq \beta_{\{u,v\}} \ . \tag{18}$$

Clearly, the optimal value of the dual program is equal to the optimal value $h'(G)$ of the primal program.

**Theorem 3.** *There is an adversarial strategy that achieves at least $\frac{1}{h'(G)}$ payoff for the adversary (regardless of the strategy of the operator).*

*Proof.* Our goal is to find a distribution $\overline{\boldsymbol{\beta}}$ such that

$$\forall T \in \mathcal{T} : \sum_{e \in E(T)} \overline{\beta}_e \lambda(T, e) \geq \frac{1}{h'(G)} \ . \tag{19}$$

Equivalently, we have to find weights $\boldsymbol{\beta} \geq \mathbf{0}$ such that

$$\forall T \in \mathcal{T} : \sum_{e \in E(T)} \beta_e \lambda(T, e) \geq 1 \tag{20}$$

and

$$\sum_{e \in E} \beta_e = h'(G) \ . \tag{21}$$

We claim that the weights $\beta_e$ of the optimal solution of the above dual linear program are such. To prove this, let $T$ be an arbitrary spanning tree and $r$ be the root of a reverse arborescence defined in Lemma 1. Then,

$$\sum_{e \in E(T)} \beta_e \lambda(T, e) = \sum_{e \in E(T)} \beta_e \lambda_r(T, e) \tag{22}$$

$$= \sum_{v \in V(G) \setminus \{r\}} \left( \sum_{e \in \{\text{edges of the } (v,r) \text{ path in } T\}} \beta_e \right) \tag{23}$$

$$\geq \sum_{v \in V(G) \setminus \{r\}} \pi_r(v) \tag{24}$$

$$\geq 1 \ , \tag{25}$$

where (22) holds by definition (see proof of Theorem 1). In (23), we used the observation that $\lambda_r(T, e)$ is the number of nodes $v$ from which the path to $r$ in $T$ contains $e$. (24) follows from Constraint 18 by applying it to every edge along the path. Finally, (25) follows from Constraint 15. □

Furthermore, the dual linear programming problem can be also solved in strongly polynomial time as the constraint matrix consists of only values of $-1$, 0 and 1. Thus, an optimal adversarial strategy can be obtained in strongly polynomial time.

## 6 Nash Equilibria and Sets of Critical Edges

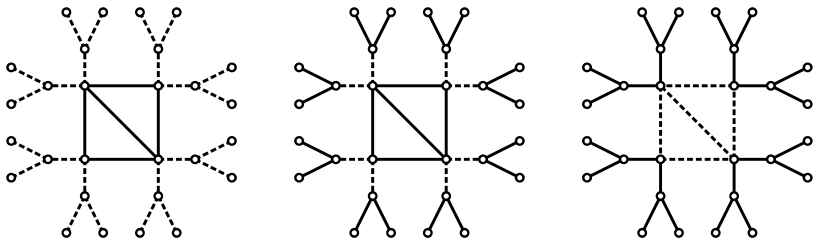
From Theorem 1 and Theorem 3, the following corollary directly follows:

**Corollary 1.** *In every Nash equilibrium, the expected payoff for the adversary (or the expected loss of the operator) is* $\frac{1}{h'(G)}$*. The optimal operator and optimal adversarial strategies form Nash equilibira of the game.*

The higher the value of $\frac{1}{h'(G)}$ is, the more vulnerable the network is. Consequently, $h'(G)$ and $\frac{1}{h'(G)}$ can be used as measures of network robustness and network vulnerability. From Theorem 2, it readily follows that these metrics can be computed efficiently.

### 6.1 Sets of Critical Edges

Besides measuring the robustness of network topologies, the network blocking game can be also used to identify critical edges that are likely to be attacked. Formally, an edge is called *critical* if it is in the support of an optimal adversarial strategy [5]. In this subsection, we make a comparison between the sets of critical edges resulting from different loss-in-value functions.



(a) GWA (b) Reed (c) BOT, Metcalfe and Linear

Fig. 3: The set of critical edges for different loss-in-value functions. Critical edges are represented by dashed lines.

In [1], the proposed loss-in-value functions were compared using the example network shown in Figure 3. For the sake of consistency, we use the same network for our comparison. The sets of critical edges identified using the previously proposed loss functions are taken from [1].

Figure 3a shows the set of critical edges identified using the simple indicator function $1_{e \in T}$ (termed GWA). The critical set consists exclusively of bridges, edges whose removal disconnects the network. This can be explained by the fact that the indicator function does not take into account the magnitude of the damage; therefore, the adversary maximizes solely the probability of hitting the spanning tree, regardless of the expected number of nodes cut off.

Figure 3b shows the set of critical edges identified using the Reed loss function. The set is similar to the one identified using the indicator function, but contains only those bridges that cut off more than one node. This result is consistent with Figure 1, which also shows that the Reed and the indicator functions are similar, but that the Reed function also takes the magnitude of the damage into account to some extent.

Finally, Figure 3c shows the set of critical edges identified using the BOT, Metcalfe and our linear loss function. Again, the fact that these three functions result in the same set is consistent with our earlier comparison based on Figure 1, which also showed that these functions are similar.

## 7 Properties of the Optimal Adversarial Strategies

In this section, we discuss properties of the optimal solutions of the dual problem. These properties allow us to formulate the dual problem as a graph partitioning problem at the end of this section. Please note that, since optimal strategies are normalized optimal solutions, Lemma 3 and Lemma 5 can be applied to optimal adversarial strategies as well.

For any $\beta \in \mathbb{R}_{\geq 0}^{|E(G)|}$, let $E(\beta) = \{e \in E(G) : \beta_e > 0\}$. Let the partitioning defined by $\beta$ be such that two nodes belong to the same partition iff there is a path between them consisting exclusively of edges in $E(G) \setminus E(\beta)$.

**Lemma 2.** *Let $V_1, \ldots, V_k$ be the partitioning defined by an optimal solution $\beta^*$ of the dual problem. If $u, v \in V_i$, then $\pi_r(u) = \pi_r(v)$ for every $r$.*

*Proof.* If $u$ and $v$ are connected by an edge $e \in E(G) \setminus E(\beta^*)$, then $\pi_r(u) = \pi_r(v)$ as $|\pi_r(u) - \pi_r(v)| \leq \beta_e^* = 0$. If $u$ and $v$ are not connected by an edge, then there has to be a path $(u, w_1), (w_1, w_2), \ldots, (w_l, v)$ consisting of edges in $E(G) \setminus E(\beta)$. Then, $\pi_r(u) = \pi_r(w_1) = \pi_r(w_2) = \ldots = \pi_r(w_l) = \pi_r(v)$ using the same argument. □

**Lemma 3.** *If $\beta^*$ is an optimal solution of the dual problem, then every edge in $E(\beta^*)$ connects nodes from different partitions defined by $\beta^*$.*

*Proof.* Assume that the claim of this lemma does not hold for a graph $G$ and an optimal solution $\beta^*$. Let $e^* \in E(\beta^*)$ be an edge that connects two nodes $u$, $v$ from the same partition. From Lemma 2, we have that $\pi_r(u) = \pi_r(v)$ for every $r$. Let $\beta' \in \mathbb{R}_{\geq 0}^{|E(G)|}$ be the following vector: $\beta'_e = \beta^*_e$ if $e \neq e^*$, and $\beta'_e = 0$ if $e = e^*$. Then,

- Constraint 18 of the dual problem is satisfied by $\beta'$, since $0 = |\pi_r(u) - \pi_r(v)| \leq \beta'_{e^*} = 0$, and no other constraint depends on the value of $\beta'_{e^*}$. Thus, $\beta'$ is a solution of the dual problem.
- The total weight of $\beta'$ is less than the total weight of $\beta^*$ as $\sum_{e \in E(G)} \beta'_e = \left(\sum_{e \in E(G)} \beta^*_e\right) - \beta^*_{e^*}$.

Therefore, $\beta^*$ cannot be an optimal solution. □

**Lemma 4.** *Let $V_1, \ldots, V_k$ be the partitioning defined by an optimal solution $\beta^*$ of the dual problem. If $r, v \in V_i$, then $\pi_r(v) = 0$.*

*Proof.* We have $\pi_r(r) = 0$ by definition. From Lemma 2, we also have that $\pi_r(v) = \pi_r(r)$ as $r$ and $v$ are in the same partition. □

**Lemma 5.** *Let $V_1, \ldots, V_k$ be the partitioning defined by an optimal solution $\beta^*$ of the dual problem and let $E(V_i, V_j)$ denote the set of edges between $V_i$ and $V_j$. For every $V_i$ and $V_j$, if $e', e'' \in E(V_i, V_j)$, then $\beta^*_{e'} = \beta^*_{e''}$ .*

*Proof.* Assume that the claim of this lemma does not hold for a graph $G$, an optimal solution $\beta^*$ and a pair of edges $e' = (v'_i, v'_j)$, $e'' = (v''_i, v''_j)$, i.e., $\beta^*_{e'} > \beta^*_{e''}$. From Lemma 2, we have that $\pi_r(v'_i) = \pi_r(v''_i)$ and $\pi_r(v'_j) = \pi_r(v''_j)$ for every $r$. Therefore, $|\pi_r(v'_i) - \pi_r(v'_j)| = |\pi_r(v''_i) - \pi_r(v''_j)| \leq \beta^*_{e''}$. Let $\beta' \in \mathbb{R}^{|E(G)|}_{\geq 0}$ be the following vector: $\beta'_e = \beta^*_e$ if $e \neq e'$, and $\beta'_e = \beta^*_{e''}$ if $e = e'$. Then,

- Constraint 18 of the dual problem is satisfied by $\beta'$, since $|\pi_r(v'_i) - \pi_r(v'_j)| \leq \beta^*_{e''} = \beta'_{e'}$, and no other constraint depends on the value of $\beta'_{e'}$. Thus, $\beta'$ is a solution of the dual problem.
- The total weight of $\beta'$ is less than the total weight of $\beta^*$ as it was decreased by $\beta^*_{e'} - \beta^*_{e''} > 0$.

Therefore, $\beta^*$ cannot be an optimal solution. □

**Lemma 6.** *There is an optimal solution of the dual problem such that if $r, s \in V_i$ then $\pi_r(v) = \pi_s(v)$ for every $v$.*

*Proof.* Let $\beta^*$ and $\pi^*$ be an optimal solution such that there exists a pair of nodes $r, s \in V_i$ that do not satisfy the constraint of the lemma, i.e, there exists an $v \in V(G)$ such that $\pi^*_r(v) \neq \pi^*_s(v)$. Let $\pi'$ be the following potential function: $\pi'_r = \pi^*_s$ and $\pi'_u = \pi^*_u$ if $u \neq r$. Since $\pi^*_s$ satisfies every constraint of the dual problem, so does $\pi'_r$; thus, $\pi'$ is also an optimal solution. By repeatedly applying the above step, we can construct a solution that satisfies the constraint of the lemma. □

From the above lemmas, the following corollary directly follows:

**Corollary 2.** *The dual problem is equivalent to the following optimization problem:*

*On every partition $V_1, \ldots, V_k$ ($k \geq 2$) of $V(G)$ and every potential function $\pi_{V_i}(V_j) \geq 0$, $\forall 1 \leq i, j \leq k$ such that*

$$\forall i: \ \pi_{V_i}(V_i) = 0 \ , \tag{26}$$

$$\forall i: \ \sum_{1 \leq j \leq k} |V_j| \cdot \pi_{V_i}(V_j) \geq 1 \ , \tag{27}$$

*minimize the objective function*

$$\sum_{1\leq i<j\leq k} |E(V_i, V_j)| \cdot \max_{1\leq r\leq k} |\pi_{V_r}(V_i) - \pi_{V_r}(V_j)| \ , \tag{28}$$

*where $E(V_i, V_j)$ is the set of edges between $V_i$ and $V_j$.*

Formulating the dual problem as a graph partitioning problem allows us to prove Theorem 4 in Section 8, which shows that $h'(G)$ is related to a graph-theoretic metric.

## 8 Relation to the Cheeger Constant

In [5], it was shown that, in case of the simple loss-in-value function $1_{e\in T}$, the payoff in every Nash equilibrium of the game is the reciprocal of the *strength* of the network $\sigma(G)$. In [7], it was shown that, in case of a natural loss-in-value function for many-to-one networks, the payoff in every Nash equilibrium is the reciprocal of the *persistence* of the network $\pi(G)$. These results link the graph-theoretic robustness of a network to game theory, which gives a better understanding of network robustness. The question naturally arises: can the above computed equilibrium payoff $\frac{1}{h'(G)}$ be linked to an elementary graph metric? In this section, we show that this is possible indeed by studying the relationship between the equilibrium payoff $\frac{1}{h'(G)}$ and the Cheeger constant $h(G)$.

In graph theory, the *Cheeger constant*[10,11] (also called *edge expansion coefficient*[12,13] or *isoperimetric number*[14,15] ) of a graph is a measure of "bottleneckedness". It is related to the spectral (or eigenvalue) gap of graph by the Cheeger inequalities and also has interesting applications, such as spectral clustering [16].

**Definition 3 (Cheeger constant).** *The* Cheeger constant *of a graph $G$, denoted by $h(G)$, is*

$$h(G) = \min\left\{ \frac{|\partial U|}{|U|} : U \subset V(G), 0 < |U| \leq \frac{|V(G)|}{2} \right\} \ , \tag{29}$$

*where $\partial U$ is the collection of all edges between $U$ and $V(G) \setminus U$.*

If $h(G)$ is low, then there is a relatively small set of edges $A$ that partitions the graph into two connected components which are both relatively large, i.e., $A$ is a "bottleneck". The intuition is that these bottlenecks correspond to the optimal attacks against a network. We will see that this is indeed true for many graphs [6].

**Theorem 4.** *For every graph $G$,*

$$h'(G) \leq h(G) \ . \tag{30}$$

[6] As a first example, we note that it is true for the network shown in Figure 3.

*Proof.* We show that the value of the optimization problem in Corollary 2, which is equal to $h'(G)$, is upper bounded by $h(G)$. Consider a restricted optimization problem, where the search space is restricted to partitions into two parts, denoted by $V_1$ and $V_2$. Since this is a minimization problem, the value of the restricted problem is an upper bound of the value of the original problem. The optimal values of the potential function are determined by the sizes of $V_1$ and $V_2$: $\pi_{V_1}(V_2) \geq \frac{1}{|V_2|}$ and $\pi_{V_2}(V_1) \geq \frac{1}{|V_1|}$. Without loss of generality, let $|V_1| \leq |V_2|$. Then, the value of the restricted optimization problem is

$$\min_{V_1 \subset V(G)} |E(V_1, V_2)| \cdot \max\{\pi_{V_1}(V_2), \pi_{V_2}(V_1)\} \tag{31}$$

$$= \min_{V_1 \subset V(G)} |E(V_1, V_2)| \cdot \max\{\frac{1}{|V_2|}, \frac{1}{|V_1|}\} \tag{32}$$

$$= \min_{V_1 \subset V(G)} |E(V_1, V_2)| \cdot \frac{1}{|V_1|} \tag{33}$$

$$= \min_{V_1 \subset V(G)} \frac{|\delta V_1|}{|V_1|} \tag{34}$$

$$= h(G) \ . \tag{35}$$

Therefore, we have that

$$h'(G) \leq h(G) \ . \tag{36}$$

□

The proof of the above theorem shows that the robustness metric $h'(G)$ can be interpreted as a possible "generalization" of the Cheeger constant to arbitrary partitionings.

**Theorem 5.** *There is a graph $G$ such that $h'(G) < h(G)$.*

*Proof.* Consider the complete graph $K_3$. It is easy to see, that the Cheeger constant of $K_3$ is $h(K_3) = 2$. We now show that the adversary can achieve a higher payoff than $\frac{1}{h(K_3)} = \frac{1}{2}$. Let the strategy of the adversary be $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. In any pure strategy of the operator, two edges are used and the expected loss of both edges is 1, since one node is cut off by the removal of each edge; therefore, the expected payoff is $\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 0 = \frac{2}{3}$. Since this is true for every pure strategy of the operator, it is also true for every mixed strategy. □

The following theorem shows that the bound is tight:

**Theorem 6.** *There are an infinite number of graphs such that $h'(G) = h(G)$.*

*Proof.* Consider a complete graph $K_{2n}$, $n \in \mathbb{Z}_+$. It is well-known that the Cheeger constant of a complete graph $K_{2n}$ is $h(K_{2n}) = \lceil \frac{2n}{2} \rceil = n$. We now show that $h'(G) \geq h(G)$ by describing an operator strategy that achieves $\frac{1}{n}$ expected payoff. Let the strategy of the operator be the uniform distribution on the set consisting of every star subgraph $S_{2n}$ of $K_{2n}$. There are $2n$ such stars; therefore, the probability of each star is $\frac{1}{2n}$. Each edge of the graph is contained

by two stars and the loss of an edge is 1 in both stars, since one node is cut off by the removal of the edge in both stars. Thus, its expected loss is $2 \cdot \frac{1}{2n} \cdot 1 = \frac{1}{n}$. Since the expected loss every edge is $\frac{1}{n}$, so is the expected payoff. □

## 9 Generalization to Non-uniform Node Weights

By measuring the *number of nodes* that are cut off from the larger connected component, we assume that each node is equally valuable or important. In practice, however, this assumption does not always hold. To relax this assumption, in this section, we generalize our results to the case where nodes have non-uniform value or importance, which can be represented by assigning a $d_v$ weight to each node.

Let $\lambda(T, e)$ measure the *total weight of nodes* that are separated from the larger connected component of the network after the attack, i.e.,

$$\lambda(T,e) := \begin{cases} \sum_{v \in \text{ smaller component of } G[T \setminus \{e\}]} d_v, & \text{if } e \in T \\ 0, & \text{if } e \notin T \end{cases} . \tag{37}$$

In this model, an optimal operator strategy is given by the following linear program:

Variables:

$$\forall r \in V(G): \ \alpha_r \in \mathbb{R}_{\geq 0}$$
$$\forall r \in V(G)\ \forall \{u,v\} \in E(G): \ f_r(u,v), f_r(v,u) \in \mathbb{R}_{\geq 0}$$

Objective:

$$\text{maximize} \sum_{r \in V(G)} \alpha_r \tag{38}$$

Constraints:

$$\forall \{u,v\} \in E(G): \sum_{r \in V(G)} f_r(u,v) + f_r(v,u) \leq 1 \tag{39}$$

$$\forall r \in V(G)\ \forall v \in V(G) \setminus \{r\}: \sum_{\{u,v\} \in E(G)} f_r(v,u) - f_r(u,v) \geq \alpha_r \cdot d_v \tag{40}$$

and an appropriately modified flow decomposition algorithm. The details of this modified algorithm can be found in Appendix A.1. Please note that the definition of the function $\lambda_r(T, e)$ is also modified appropriately: it measures the total weight of nodes that are separated from $r$ in $G[T \setminus \{e\}]$.

An optimal adversarial strategy is given by the dual problem:

Variables:

$$\forall e \in E(G): \ \beta_e \in \mathbb{R}_{\geq 0}$$
$$\forall r, v \in V(G), r \neq v: \ \pi_r(v) \in \mathbb{R}_{\geq 0}$$

Objective:

$$\text{minimize} \sum_{e \in E(G)} \beta_e \tag{41}$$

Constraints:

$$\forall r \in V(G): \sum_{v \in V(G) \setminus \{r\}} \pi_r(v) \cdot d_v \geq 1 \tag{42}$$

$$\forall r \in V(G)\ \forall \{u,v\} \in E(G): \ |\pi_r(u) - \pi_r(v)| \leq \beta_{\{u,v\}} \ . \tag{43}$$

Otherwise, everything is the same. Since the proofs for this model can be obtained by appropriately modifying the original proofs in a very straightforward way, we omit them here.

## 10 Conclusions

In this paper, we introduced a linear loss-in-value function for the network blocking game. As one of our main contributions, we provided strongly polynomial-time algorithms to compute optimal adversarial and operator strategies and, thus, the payoff in the Nash equilibria of the game. To the best of our knowledge, these are the first efficient algorithms for the network blocking game with a loss-in-value function that is not too simplistic. The efficiency of these algorithms allows us to measure the game-theoretic robustness of networks in practice. Furthermore, the optimal strategies can be also used to identify critical edges that are likely to be attacked. We also generalized our model to non-uniform node weights, which allows nodes to have varying importance or value.

In addition, we proved that the payoff in the Nash equilibria of the game is closely related to the Cheeger constant of a graph (also called minimum edge expansion or isoperimetric number), a well-known metric in graph theory. Therefore, the game-theoretic robustness metric resulting from the linear loss-in-value function can be related to a graph-theoretic robustness notion.

## Acknowledgements

This paper has been supported by HSN Lab, Budapest University of Technology and Economics, `http://www.hsnlab.hu`. Levente Buttyán is supported by the Hungarian Academy of Sciences through the funding of the Academic Research Group on Information Systems (ID: 04-130). The work is also related to the internal project of the authors' hosting institution on "Talent care and cultivation in the scientific workshops of BME", which is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009.

## References

1. Gueye, A., Marbukh, V., Walrand, J.C.: Toward a metric for communication network vulnerability to attacks: A game theoretic approach. In: Proc. of the

3rd International ICST Conference on Game Theory for Networks. GameNets'12, Vancouver, Canada (May 2012)
2. Cunningham, W.: Optimal attack and reinforcement of a network. Journal of the ACM **32**(3) (1985) 549–561
3. Bauer, D., Broersma, H., Schmeichel, E.: Toughness in graphs–a survey. Graphs and Combinatorics **22**(1) (2006) 1–35
4. Laszka, A., Buttyán, L., Szeszlér, D.: Optimal selection of sink nodes in wireless sensor networks in adversarial environments. In: Proc. of the 12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia. WoWMoM'11, Lucca, Italy (June 2011) 1–6
5. Gueye, A., Walrand, J.C., Anantharam, V.: Design of network topology in an adversarial environment. In: Proc. of the 1st International Conference on Decision and Game Theory for Security. GameSec'10, Berlin, Germany (November 2010) 1–20
6. Gueye, A., Walrand, J.C., Anantharam, V.: A network topology design game: How to choose communication links in an adversarial environment? In: Proc. of the 2nd International ICST Conference on Game Theory for Networks. GameNets'11, Shanghai, China (April 2011)
7. Laszka, A., Szeszlér, D., Buttyán, L.: Game-theoretic robustness of many-to-one networks. In: Proc. of the 3rd International ICST Conference on Game Theory for Networks. GameNets'12, Vancouver, Canada (May 2012)
8. Schwartz, G., Amin, S., Gueye, A., Walrand, J.: Network design game with both reliability and security failures. In: Proc. of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE (September 2011) 675–681
9. Tardos, E.: A strongly polynomial algorithm to solve combinatorial linear programs. Operations Research **34**(2) (1986) 250–256
10. Chung, F.R.K.: Spectral graph theory. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, Providence, R.I. (1997)
11. Chung, F.R.K.: Laplacians and the Cheeger inequality for directed graphs. Annals of Combinatorics **9**(1) (2005) 1–19
12. Alon, N.: On the edge-expansion of graphs. Combinatorics, Probability and Computing **6**(2) (1997) 145–152
13. Alon, N.: Spectral techniques in graph algorithms. In: Proc. of the 3rd Latin American Symposium on Theoretical Informatics, Campinas, Brazil (April 1998) 206–215
14. Mohar, B.: Isoperimetric numbers of graphs. Journal of Combinatorial Theory, Series B **47**(3) (1989) 274–291
15. Mohar, B.: Isoperimetric inequalities, growth, and the spectrum of graphs. Linear Algebra and Its Applications **103** (1988) 119–131
16. Bühler, T., Hein, M.: Spectral clustering based on the graph p-Laplacian. In: Proc. of the 26th Annual International Conference on Machine Learning. ICML'09, Montreal, Canada (June 2009) 81–88

## A Flow-decomposition Algorithm

We introduced our flow decomposition algorithm in [7]. To make this paper self-contained, in this section, we give an overview of the algorithm. For a detailed proof, see [7].

In Section 4, we used two non-negative flow variables for each undirected edge to allow the summation of the absolute values of multiple flows in the linear program. For simplicity, in the following description, we use a more classical approach and replace each edge with two arcs facing opposite directions, each arc having a single non-negative flow value.

**Theorem 7.** *Given a graph $G$ with a sink node $r$ and a multi-source flow $f$* [7] *such that each $v \in V(G) \setminus \{r\}$ node is a source producing $\alpha$, there exist weights $\alpha_T$, $T \in \mathcal{T}(G)$ such that $\sum_{T \in \mathcal{T}(G)} \alpha_T = \alpha$ and $\forall e \in E(G) : \sum_{T \in \mathcal{T}(G)} \alpha_T \cdot \lambda_r(T, e) \leq f(e)$* [8] *.*

*Proof.* Our proof is constructive and it is based on the following algorithm:

1. Find a spanning reverse arborescence $T$ rooted at $r$ in $G$ such that
   - $T$ only includes edges to which $f$ assigns a positive flow amount and
   - every edge is directed in the same way as the flow.
2. Calculate $\lambda_r(T, e)$ for every $e \in T$.
3. Let $\alpha_T := \min_{e \in T} \frac{f(e)}{\lambda_r(T,e)}$.
4. For every $e \in E(G)$, let $f(e) := f(e) - \alpha_T \cdot \lambda_r(T, e)$.
5. If the incoming flow assigned by $f$ to $r$ is greater than zero, then continue from Step 1.
6. Let $\alpha_T := 0$ for every other spanning tree.

Before proving the correctness of the algorithm, we have to prove that Step 1 can be executed in each iteration, otherwise the algorithm would terminate with an error. Obviously, if $f$ is a network flow and the amount of outgoing flow from every $v \in V(G) \setminus \{r\}$ is positive, there has to be a directed path from every $v \in V(G) \setminus \{r\}$ to $r$ consisting of edges with positive flow amounts. Thus, we have to show that the outgoing flow from every $v \in V(G) \setminus \{r\}$ remains positive as long as the incoming flow to $r$ is positive.

For a $v \in V(G) \setminus \{r\}$, let $\Lambda_v$ denote $\lambda_r(T, e_{out})$, where $e_{out}$ is the outgoing edge of $v$ in $T$. Clearly, the sum of $\lambda_r(T, e_{in})$ over all incoming edges $e_{in} \in E(G)$ of $v$ is $\Lambda_v - 1$. Since the flow along every edge $e$ is decreased by $\alpha_T \cdot \lambda_r(T, e)$, the sum of outgoing flows is decreased by $\alpha_T \cdot \Lambda_v$. Similarly, the sum of incoming flows is decreased by $\alpha_T \cdot (\Lambda_v - 1)$. Therefore, the net outgoing flow of $v$ is decreased by $\alpha_T$. Since the outgoing flow of every $v$ is the same at the beginning and it is decreased by the same amount in every iteration, they are decreased to zero simultaneously.

Now, we can prove the correctness of the algorithm. First, we have to prove that $\alpha$ is indeed a distribution. This is evident, as the amount of incoming flow to $r$ is decreased by $\alpha_T(|V(G)| - 1)$ at every assignment, and the amount is $|V(G)| - 1$ at the beginning and zero after the algorithm has finished; therefore, $\sum_{T \in \mathcal{T}} \alpha_T = 1$.

[7] A multi-source flow is a network flow with a set of sources instead of only one source.

[8] Please recall the definition of $\lambda_r(T, e)$ from Section 4: it is the number of nodes that are separated from $r$ in $G[T \setminus \{e\}]$.

Second, we have to prove that $\forall e \in E(G) : \sum_{T \in \mathcal{T}(G)} \alpha_T \cdot \lambda_r(T, e) \leq f(e)$. At every $\alpha_T$ assignment, the flow along every edge is decreased by $\alpha_T \cdot \lambda_r(T, e)$ and it is never decreased to a negative value. Therefore $\sum_{T \in \mathcal{T}} \alpha_T \cdot \lambda_r(T, e) \leq f(e)$.

Finally, we show that the algorithm terminates after at most $|E(G)|$ iterations. In every iteration, the flow along at least on edge (i.e., along every edge for which $\frac{f(e)}{\lambda_r(T,e)}$ is minimal) is decreased from a positive amount to zero. Since there are $|E(G)|$ edges, there can be at most $|E(G)|$ iterations. □

From the last paragraph of the proof, it also follows that the support of the resulting distributions consists of at most $|E(G)|$ spanning trees.

### A.1 Flow-decomposition Algorithm with Non-uniform Node Weights

The algorithm is fundamentally the same as in the case of uniform node weights. The following modifications have to be made:

- Each $v \in V(G) \setminus \{r\}$ node is a source producing $\alpha \cdot d_v$, instead of $\alpha$.
- Consequently,
  - the sum of $\lambda_r(T, e_{in})$ over all incoming edges $e_{in} \in E(G)$ of $v$ is $\Lambda_v - d_v$, instead of $\Lambda_v - 1$,
  - the net outgoing flow of $v$ is decreased by $\alpha_T \cdot d_v$, instead of $\alpha_T$,
  - the incoming flow to $r$ is decreased by $\alpha_T \sum_{v \in V(G) \setminus \{r\}} d_v$, instead of $\alpha_T(|V(G)| - 1)$.