Center for Economic Institutions

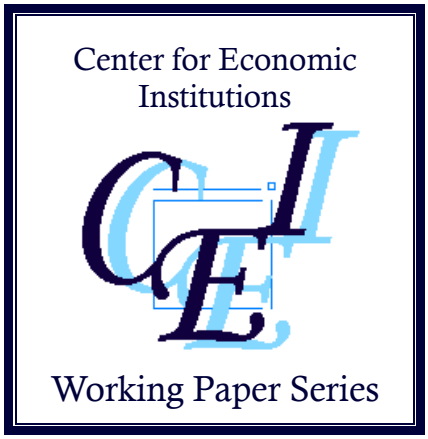
Working Paper Series

No. 2015-2

“From Sabotage Games to Border Protection”

Dmitriy Kvasov

June 2015



Institute of Economic Research
Hitotsubashi University
2-1 Naka, Kunitachi, Tokyo, 186-8603 JAPAN
http://cei.ier.hit-u.ac.jp/English/index.html
Tel:+81-42-580-8405/Fax:+81-42-580-8333

From Sabotage Games to Border Protection*

Dmitriy Kvasov†

June 2015

Abstract

Sabotage games on a graph involve Runner who wants to travel between two given vertices and Blocker who aims to prevent Runner from arriving at his destination by destroying edges. This paper introduces and studies several generalizations of sabotage games. First, it completely characterizes games with multiple destinations on weighted trees for both local and global cutting rules of arbitrary capacity, using an algorithmic labeling procedure. Second, it introduces the transformation procedure that associates a weighted tree with any weighted graph. The procedure allows complete characterization of games on weighted graphs for local cutting rules of arbitrary capacity and provides sufficient conditions for Blocker to win for global cutting rules. The applications of sabotage games to the issue of border security are discussed.
Keywords: sabotage games, games on graphs, dynamic graph reliability, network interdiction, border security
JEL codes: C720, C730, D850

1 Introduction

On October 29, 2011 New York City has experienced an unusually early and heavy snowstorm (2011 Halloween nor'easter). The storm left more than three million people without power and shut down critical landing systems at New York's major airports, disrupting air travel and forcing more than 150 flights to divert.

Among them was the JetBlue Flight 504 from Fort Lauderdale. Flight 504's pilot has asked to land in Boston. But Boston was soon swamped with two dozen diverted planes, some parked directly on the runways, closing it other landings, there was no place for Flight 504 anymore. The air transportation network literally started to crumble under Flight 504's wings. It was then re-directed to land in Hartford, but that airport was also directly in the snowstorm's path. Flight 504 has been able to land but the airport's gates were already overcrowded, weather has worsened, and power outages were constant.

*I am very grateful to the faculty and staff of CEI at Hitotsubashi University for their hospitality during my stay in January-April 2015.
†University of Adelaide, Adelaide, SA 5005, Australia, e-mail: kvasov.dmitriy@gmail.com

After the three-hour flight 129 passengers and 6 crew members of Flight 504 has been forced to spend additional seven and a half hours on a tarmac without water and food.[1]

The travails of hapless Flight 504 epitomize a pervasive problem: traversing a network while it is falling apart. The dual problem of restricting or blocking undesirable traffic in a given network is as widespread. This paper aims to provide a comprehensive game-theoretic framework suitable for the analysis of the general issue of dynamic network reliability or vulnerability.

The scope for potential applications of the developed framework is very large. It can be used to study a wide variety of issues, including design of reliable networks for transportation, information exchange, computation, etc., as well as the dual problem of dismantling unwanted networks, or preventing the propagation of undesirable traffic of any kind in a given network. More specifically, the framework can be used to study, in a dynamic setting, positioning of the traffic control units; interception of drug traffic or smuggling; or routing of information in presence of the malicious opponent who wants to stop the communication. Section 6 discusses how the results of the paper can be applied to the specific question of border security.

In this paper we model the problem of network traversing in presence of adversary as a sequential move game with two players. One player, Runner, wants to travel along the edges of the given graph from a given starting node to his destination node. The other player, Blocker, tries to prevent Runner from arriving at his destination by blocking or destroying edges in the graph.[2] The class of games studied in the paper is referred to as sabotage games.[3]

The example in Figure 1 illustrates the main elements of the game. Consider a multigraph—that is, a graph with possible multiple edges between the vertices—with five vertices

$$\{a, b, l, k, s\}$$

and edges

$$\{ab_1, ab_2, al_1, al_2, ak, bl, lk, bs, ls_1, ls_2, ks\}.$$

Runner, starts at vertex s[4] and wants to arrive at his destination, vertex a, traveling one edge at a time. Blocker destroys edges, also one at a time, aiming to prevent Runner from reaching a. Players alternate, starting with Blocker, and the current state of the transportation network is common knowledge at all times.

Starting from the vertex s, Runner has a winning strategy. Blocker's opening move may block b (by deleting bs) or k (by deleting ks), but Blocker cannot prevent Runner

[1] The outline of the Flight 504 incident follows Gerchick (2013).

[2] If, as in example of Flight 504, the role of Blocker is played by Nature then solving the game provides the characterization of the worst-case scenario.

[3] The terminology is adopted from van Benthem (2005) who pioneered the use of the similar games in computer science literature.

[4] The example is from van Benthem (2005) where protagonists are Logician and Demon. Logician from Saarbrüken wants to go to a conference in Amsterdam, hence the names of the vertices.

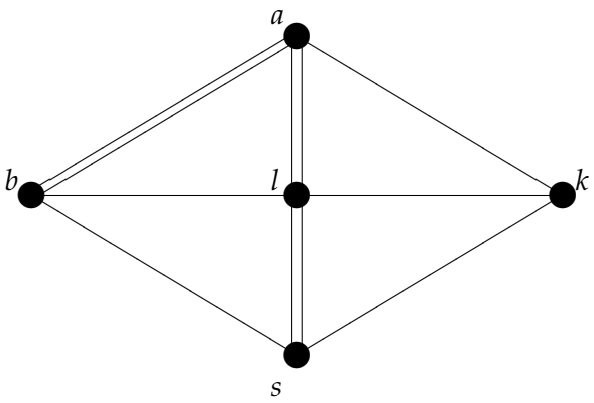


Figure 1: A sabotage game.

from getting first to l, and then to a. Blocker may also start by cutting one of the edges between a and l—to prepare a trap—but in such case Runner replies by going to b first. In the same network, however, if Runner needs to go back from a to s, Blocker has the winning strategy by first destroying one of the edges between l and s.

Next, we discuss informally the main elements of the sabotage game as studied in this paper, to provide context and intuition for formal definitions in Section 3.

The graph on which the game is played is referred to as the sabotage graph. The sabotage graph can be either a graph or a weighted graph (as in Figure 1). In the latter case an edge is characterized by an integer capacity (or weight) that is interpreted as a number of different direct routes between two vertices. A subclass of (weighted) sabotage graphs, namely (weighted) trees, is analysed separately. To make games on trees non-trivial we allow Runner to have more than one destination.

The set of actions available to Blocker as the game progresses is referred to as cutting rules. It is important to distinguish between the "local" and the "global" cutting rules. We say that Blocker obeys the local cutting rule if the edges Blocker is able to delete at her move are only those incident with the current vertex of Runner. In contrast, the global cutting rule allows Blocker to delete an edge anywhere in the graph, independent of the current position of Runner.

Similarly, the set of actions available to Runner as the game evolves is referred to as moving rules. The "local" Runner can only travel from his current vertex to an adjacent one. The global movement rule would allow Runner to "secure" an edge anywhere in the graph with the provision that once secured, the edge cannot be destroyed by Blocker anymore. Runner wins if he is able to secure a path from the origin vertex to the destination vertex.

Another important generalization of the cutting rule, novel to this paper, is that we allow Blocker to cut more than one edge at a time, or, more precisely, Blocker is characterized by a "capacity of a cut", that is the total capacity of the edges she is

allowed to cut at any of her moves. As an example, consider the sabotage graph in Figure 1 with local Blocker who has cut capacity 2 and Runner at s. In such game, Runner does not stand a chance. Blocker could start by deleting the edges bs and ks, forcing Runner to move to l. At her next move Blocker cuts both edges between l and a, and so on.

The first main contribution of the paper is the complete characterization of sabotage games with multiple destinations on weighted trees for both local and global cutting rules of arbitrary capacity. The paper introduces an algorithmic labeling procedure that determines the winner of any sabotage game on a weighted graph.

The second main contribution of the paper is the transformation procedure that associates a weighted tree to any weighted graph, thus reducing the sabotage games on graphs to sabotage games on trees. This transformation can be viewed as a simplification of the original graph which by getting rid of the cycles 'straightens' all the paths to destinations. Such transformation allows to completely characterize sabotage games on weighted graphs in case of local cutting rule of arbitrary capacity. In case of global cutting rule, however, sabotage games on graphs and trees are not necessarily equivalent. The transformation of the graph into tree restricts strategies available to Blocker and makes it more difficult for her to win. The associated game on the weighted tree provides sufficient but no longer necessary condition for Blocker to win.

The rest of the paper organized as follows. Section 2 reviews the related literature. Section 3 defines the class of sabotage games. Section 4 characterizes the sabotage games on weighted trees both with local and global Blocker. Section 5 introduces the transformation of weighted graphs into weighted trees and characterizes games on weighted graphs. Section 6 provides application of sabotage games to the question of border security. Section 7 concludes.

2 Related Literature

The problem of dynamic graph reliability was first introduced by Papadimitriou (1985) as a problem of decision-making under uncertainty. He considers a player who wishes to traverse the graph while it is falling apart and the probability of an edge failure before the next move of the player depends on his current position. Sabotage games have been introduced, along with the name, by van Benthem in 2002 (see van Benthem (2005)) and further studied in Löding and Rohde (2003), Rohde (2004), and Klein et al., (2012). This literature studies sabotage games from the perspective of computer science and focuses mostly on the computational complexity of the game and on its applications to logic, learning, and model-checking. This literature does not address the game-theoretical aspects of sabotage games. The closest paper is Radmacher and Thomas (2008) which also points out the difference between local and global rules and uses a game-theoretic terminology.

Another related strand of the literature is operations research literature on network interdiction (see Collado and Papp (2010) for a concise recent review). Network interdiction is the monitoring or halting of an adversary's activity on a network. Its models involve two players, usually called the interdictor and the evader. The evader operates on the network to optimize some objective such as moving through the network as fast as possible (shortest path interdiction), or with as little probability of being detected as possible (most reliable path interdiction), or to maximize the amount of goods transported through the network (network flow interdiction). The interdictor has the ability to change the structure or parameters of the network (remove vertices or edges, increase detection probabilities, or lower edge capacities) in order to minimize the evaderŠs objective function. Starting with Washburn and Wood (1995) this literature often takes the game-theoretic approach to the problem, using two-person simultaneous move zero-sum games to model network interdiction. The important differentiating feature of our paper is the use of sequential game framework which allows us to address the dynamic aspects of network interdiction.

The extensive economic literature on economic networks and network games (see Jackson (2010) and Jackson and Zenou (2015) for incisive reviews) has been successful in providing valuable insights into understanding many important economic and social activities. Hong (2011) studies a strategic network flow interdiction problem. The evader chooses a flow specifying a plan for carrying bads through a network from a base to a target. Simultaneously, the interdictor chooses a blockage specifying a plan for blocking the flow of bads through arcs in the network. McBride and Hewitt (2013) pioneered the use of games with incomplete information in network interdiction problem. They examine disruption of covert and illegal networks by an interdictor who does not have a complete knowledge of the network structure. Dell (2014) provides an empirical analysis of Mexican policy towards the drug trade using the network model of trafficking routes.

Finally, two related games of pursuit and evasion must be mentioned. The angel and the square-eater game (Berlekamp et al, 1982) might be viewed as very specific generalization of sabotage games but its solution relies on a very different techniques. An angel (of power x) can fly to any square of an infinite chessboard which could be reached by an x King moves. The square-eating devil wins if he can surround the angel with sulphurous moat, an x squares wide, of eaten squares. Another game of pursuit on graphs, the cops and robbers game was introduced by Nowakowski and Winkler (1983). A cop chooses a vertex of a given graph, then the robber chooses a vertex, then the players move alternately beginning with the cop. A move consists of staying at one's present vertex or moving to an adjacent vertex; each move is seen by both players. The cop wins if he manages to occupy the same vertex as the robber, and the robber wins if he avoids this forever.

3 Definitions

3.1 Weighted Graphs

This section provides a number of standard graph-theoretical definitions used in the paper.

A graph $G = (V, E)$ is a finite non-empty set V of *vertices* and a set E of edges that connect some of the vertices. An *edge* is an unordered pair of distinct vertices. Note that the definition allows at most one edge joining two vertices and does not allow edges that join a vertex to itself. Two vertices $v_\imath$ and $v_\jmath$ of G are *adjacent* if $v_\imath v_\jmath$ is an edge of G; and the vertices $v_\imath$ and $v_\jmath$ are *incident* with the edge $v_\imath v_\jmath$.

A (simple) *path* on the given graph $G = (V, E)$ is a sequence of distinct vertices (except possibly, the first, v_1, and the last, v_n)

$$v_1, v_2, \ldots, v_n$$

such that any two consecutive vertices in the sequence are adjacent. A *cycle* is a path with more than three vertices in which the first and the last vertices are the same. A graph is *connected* if there is a path between each pair of its vertices.

The standard notion of the graph, however, is not rich enough to describe all non-trivial networks on which the sabotage game could be played. What makes the game on the network in the Figure 1 interesting and non-trivial is the existence of multiple edges between some vertices. To formally describe networks with multiple edges the notion of a weighted graph is used.[5]

A *weighted graph* is a triple $G = (V, E, w)$, where V is a finite non-empty set of vertices, E is a set of edges, and w is a weight function that associates a non-negative integer—weight or capacity—with every edge of the graph,

$$w : V \times V \to \mathbb{N}.$$

Zero weight is interpreted as the absence of an edge between the vertices $v_\imath$ and $v_\jmath$. The weight function is assumed to be symmetric, $w(v_\imath v_\jmath) = w(v_\jmath v_\imath)$. Two vertices $v_\imath$ and $v_\jmath$ are adjacent if the weight $w(v_\imath v_\jmath) > 0$. The definitions of a path, a cycle, and connectivity for weighted graphs are the same as for graphs.

Even though the weighted graph $G = (V, E, w)$ is completely determined by its set of vertices and its weight function, it is convenient to interpret the values of the weight function as the number of edges between vertices.

Thus, for example, the transportation network in Figure 1 can be represented by the weighted graph with the vertices $V = \{a, b, l, k, s\}$ and the weights $w(ab) = w(al) = w(ls) = 2$ and $w(ak) = w(bl) = w(lk) = w(bs) = w(ks) = 1$. There is only one path between a and l, namely the path a, b, l.

[5] Alternatively, the multigraph (or pseudograph) could be used.

An important set of results in the paper is obtained for the special class of weighted graphs, that of weighted trees. A *weighted tree* is a connected weighted graph with no cycles. As a result, there is a unique path between any two vertices of the given tree. A *rooted tree* has a distinguished vertex called the root, that induces a natural partial order on vertices, away from (or towards) the root. In a rooted tree, the *parent* of a vertex is the vertex adjacent to it on the path to the root; every vertex except the root has the unique parent. A *child* of a vertex v is a vertex of which v is the parent. A *terminal vertex* (or a leaf) is a vertex that has no children. The set of terminal vertices is denoted by T.

3.2 The Game

This section formally introduces the general class of the sabotage games and discusses several variations of the rules.

A two-player sabotage game is played on the given connected weighted graph $G(V, E, w)$, referred to as the sabotage graph. Player 2, called Runner, is initially located at the *starting vertex* v_0 and wants, by traversing the graph to arrive in his *set of destinations*, D. The set of destinations may be a singleton or it may contain more than one vertex. In the latter case, Runner wins by arriving at either vertex in D.

The objective of Player 1, called Blocker, is to prevent Runner's arrival at a destination by deleting edges in the graph. Thus, Blocker wins whenever Runner is not in the destination set and there exists no path from his current position to any of the destination vertices.

Players move sequentially starting with Blocker. Blocker is endowed with the strictly positive integer *cut capacity*, c, which is the upper limit on the total weight of the edges she can delete at any one move. In addition, Blocker is subject to *cutting rules*. Under the local cutting rule, Blocker is only able to delete edges incident to the current position of Runner subject to not exceeding her cut capacity. Under the global cutting rule, Blocker is able to reduce the weight of any combination of edges anywhere on the graph but again subject to not exceeding her cut capacity. For example, a global Blocker with cut capacity of 2, can reduce the weight of any edge by two, reduce the weight of two edges by one, reduce the weight of just one edge by one, or do nothing.

When it is Runner's turn to move, Runner can travel from his current vertex v_n to any vertex v_{n+1} provided that $w(v_n, v_{n+1}) > 0$. That is Runner can travel to any vertex adjacent to the one he is currently at. After the move, Runner's current position on the graph becomes v_{n+1}. All the results in the paper are derived for the case of the local Runner[6].

Thus, every move of Runner changes his position on the graph and every move of Blocker changes the weight function of the graph. A position in the sabotage game then is a tuple (v_i, w_j), where v_i is the current vertex of Runner and w_j is the current

[6]It is also possible to consider global moving rules for Runner, such games are discussed in Section 7

weight function of the graph. The set of all legal positions (positions reached from the initial position by a sequence of feasible moves of Blocker and Runner) is denoted by $\mathcal{P}$. A play of the game is a finite sequence of positions:

$$(v_0, w_0)(v_0, w_1)(v_1, w_1)\dots$$

where $w_0 = w$.

A strategy of Blocker, given the cutting rule, is a function $b : \mathcal{P} \to w$ that maps each possible position to a weight function. A strategy of Runner is a function $r : \mathcal{P} \to V$, that maps each possible position to the vertex to which Runner moves next.

Note that without loss of generality, only memory-less strategies are considered, that is strategies that depend only on the current position in the game but not on the specific play that leads to that position.

Thus defined game is a finite two-person zero-sum game with perfect information; at any stage of the game the current position of Runner and the current state of the graph (its weight function) is a common knowledge among players.

To summarize, a two-player *sabotage game* consists of

1. a connected weighted graph $G = (V, w)$,
2. a starting vertex $v_0 \in V$,
3. a non-empty set of destinations, $D \subset V$,
4. cutting capacity, c and cutting rule for Blocker,
5. movement rule for Runner,
6. payoff function.

The solution concept is subgame perfect equilibrium.

4 Weighted Trees

This section characterizes the equilibria of the sabotage games played on the weighted trees. When the sabotage graph is a weighted tree, the starting vertex is the root of the tree and the set of destinations is the set of terminal vertices of the tree.

4.1 Simple Trees

Note that on a graph, given local cutting rule and a single destination, solving the game becomes straightforward. Blocker can lay in wait doing nothing till the Runner reaches a vertex adjacent to the destination. Then, Blocker cuts the edge in question. In this way Blocker guarantees herself a win. The solution is not as obvious if the set of destinations has more than one vertex.

To illustrate the proof techniques and to highlight the intuition behind the results I start with a simple class of the sabotage games—the games played on simple trees. The weight function on simple trees only takes values 0 and 1, the former means there

is no edge between two vertices and the latter means that there is the edge between the two. Blocker has a unary cut capacity: at every move she can delete at most one edge.

First, observe that to find the optimal strategy of Blocker it suffices to consider Blocker's local strategies, ones that utilize only the local cutting. On simple trees, Blocker who can delete only edges incident to the current position of Runner is as powerful as global Blocker.

Lemma 1. *On simple trees, for any optimal strategy of Blocker there exists a local optimal strategy.*

Suppose that at some stage in the game, Runner is at the vertex v_t and the optimal strategy of Blocker calls for deletion of an edge not incident with the current position of Runner, say, deletion of an edge at the vertex v_s. Consider a strategy of Blocker that instead removes an edge incident with v_t which is on the path to v_s. Thus modified strategy of Blocker is clearly also optimal as it reduces by one the number of moves available to Runner at v_t and coincides with the original one for all moves still available to Runner at v_t. This procedure can be repeated for all non-local moves of Blocker, reducing a global strategy of Blocker to an equivalent local one in the finite number of steps.

The second observation, also specific to the simple trees, is that it is (weakly) sub-optimal for Runner to backtrack. Runner's optimal strategy can always be formulated as the strategy without cycles. If Runner is able to reach a destination by first going along a branch of the tree then retracing part of his way and after that going along different branch to a terminal vertex, he can do as well by going to that terminal vertex directly. Thus, the interaction of Runner and Blocker on simple trees can be reduced to a series of interactions at each vertex.

It is easy to understand the mechanics of what happens at each vertex: Blocker can stop Runner at a vertex only if she is able to cut all the incident edges that lead to terminal vertices. However, in order to win the game Blocker does not necessarily need to block every path at the given vertex; she might be able to block some of those paths later.

The next two propositions characterize simple trees by aggregating local decision at each vertex into overall strategy. Proposition 1 provides a very straightforward and intuitive characterization of the sabotage games on simple trees such that every non-terminal vertex has at most two children.

Proposition 1. *On a simple tree G such that every non-terminal vertex has at most two children, Runner wins if and only if every non-terminal vertex has exactly two children.*

Proof. Suppose there is a non-terminal vertex with only one child, say vertex v_k. Consider a path from v_0 to v_k:

$$(v_0, v_1, v_2, \ldots, v_{k-1}, v_k).$$

Note that since G is a tree, this path is unique. Player 1 can force Player 2 to follow that path, thus winning the game. Indeed, at his i's move ($i = 0, 1, \ldots, k-1$), Blocker removes the edge at v_i that is not incident to v_{k+1}. If there is no such edge, Blocker removes any other edge or does nothing. Thus at every move, Runner is forced to move along the path from v_0 to v_k. Once Runner reaches v_k, Blocker removes the only remaining incident edge.

If every vertex has exactly two children[7], then Runner has a winning strategy, because after any move of Blocker, Runner can move to the vertex such that all non-terminal children of that vertex have exactly two children. In that way, Blocker cannot prevent Runner from reaching the destination set. □

An intuitive way to extend Proposition 1 to the case of arbitrary simple trees is by utilizing the notion of the induced subgraph. A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subset V$ and $E' \subset E$, and is denoted by $G' \subset G$. If G' contains all edges of G that join two vertices in V' then G' is said to be the *subgraph induced by* V'.

The Proposition 1 then becomes:

Proposition 2. *On a simple tree $G = (V, E)$ Runner wins if and only if there exists an induced subgraph $G' = (V', E')$ of G such that $v_0 \in V'$, $T \cap V' \neq \emptyset$, and every non-terminal vertex of G' has exactly two children.*

Instead of proving Proposition 2 directly we introduce a "labeling" procedure which is just thinly disguised backward induction procedure applied to the sabotage tree itself. The labeling approach then used to prove all the subsequent results for weighted trees.

The procedure labels all vertices of a tree with either $\mathcal{W}$ or $\mathcal{L}$. When Runner is at $\mathcal{W}$-vertex he is guaranteed to win the game if he plays optimally. When Runner is at $\mathcal{L}$-vertex, he looses no matter what he does if Blocker plays optimally. The vertices are labeled recursively starting with the terminal ones. All terminal vertices are labeled with $\mathcal{W}$, since reaching a terminal vertex is the winning condition for Runner. Next, all vertices whose children has been already assigned labels are, in turn, labeled. The vertex is labeled with $\mathcal{W}$ if it has more than one $\mathcal{W}$-child. It reflects the fact that at such vertex Blocker cannot prevent Runner from reaching $\mathcal{W}$-vertex on his next move. Otherwise, the vertex is labeled with $\mathcal{L}$. The labeling continues until all the vertices has been labeled. The procedure always stops after a finite number of steps since the number of vertices of sabotage graph is assumed to be finite. The procedure is well-defined—there is no ambiguity in labeling—since the label of a vertex is uniquely determined by the labels of its children. Formally, then, the labeling procedure is:

1. Label all terminal vertices with $\mathcal{W}$.

2. If a vertex has all its children labeled, label it with $\mathcal{W}$ if it has more than one $\mathcal{W}$-child, otherwise label it with $\mathcal{L}$.

3. Repeat previous step until all vertices are labeled.

[7]Such trees are often referred to as full binary trees.

Proposition 2 then takes the following form:

Proposition 3. *On a simple tree G Runner wins if and only if the root is $\mathcal{W}$-vertex.*

Proof. Given a labeled tree with the $\mathcal{W}$-root it is easy to recover a winning strategy for Runner, as all hard work was already done by labeling procedure. All that Runner needs to do at every move is to go from a $\mathcal{W}$-vertex to another $\mathcal{W}$-vertex. Blocker cannot hinder him in that, since—by construction—any $\mathcal{W}$-parent vertex has at least two $\mathcal{W}$-children vertices.

Suppose, conversely, that Runner has winning strategy, then it must be the case that any vertex that Runner traverses on his way to a destination has at least two children. Otherwise, the path through that vertex would have been blocked by Blocker. Thus, the labeling procedure would assign $\mathcal{W}$ to every non-terminal vertex on that path, up to and including the root. □

4.2 Weighted trees

The characterization of the sabotage games on the weighted trees with Blocker with cut capacity c constrained by local cutting rule is the straightforward generalization of Proposition 3 with the only modification: instead of being able to cut only one edge, Blocker is able to cut up to c incident edges.

The modified labeling procedure is:

1. Label all terminal vertices with $\mathcal{W}$.

2. If a vertex has all its children labeled, label it with $\mathcal{W}$ if the sum of the weights of edges incident to $\mathcal{W}$-children is more than c, otherwise label it with $\mathcal{L}$.

3. Repeat previous step until all vertices are labeled.

Proposition 4. *Suppose G is a weighted tree and Blocker's cut capacity is c under the local cutting rule. Runner wins if and only if the root is $\mathcal{W}$-vertex.*

The most intriguing property of the weighted trees—and the one that makes the sabotage game non-trivial—is that local and global cutting rules are not equivalent. The hinge of proof of Lemma 1 is that it is equally difficult to cut a path at any vertex, thus it can as well be done as early as possible.

On weighted trees a path may have a "weak" link; moreover, it can happen for two different reasons. On one hand, an edge is weak if it has a smaller weight than the other edges in the path. On the other hand, there is a sort of depreciation of the edges' weights along the path. Consider a tree in Figure 2. If $c = 1$, it is, in fact, the last edge in the path that is weak. Blocker can prevent Runner from arriving at his destination because Blocker can gradually destroy that last edge by the time it takes Runner to reach it. This difference is also manifest in games on weighted graphs in general, it is for this very reason Blocker is able to beat Runner in the game of Figure 1 when Runner is going from a to s.

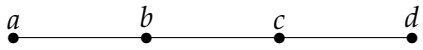


Figure 2: A weighted tree with root a and weights $w(ab) = 2$, $w(bc) = w(cd) = 3$.

For the analysis of Blocker's global cutting rules on weighted trees the labeling procedure needs a substantial modification. The main idea is to use recursion to determine—at every vertex—the minimal optimal number of cuts (the total weight of the edges) needed by Blocker to prevent Runner from winning. That number is the value of the labeling function L.

First, all terminal vertices are labeled with ∞. This is a convenient way to express the fact that once Runner reaches the terminal vertex no amount of additional cuts available to Blocker can prevent Runner from winning.

Next, consider all vertices whose only children are terminal vertices. Take any such vertex, say v_j, it has children indexed by k and the weights of corresponding edges are w_{jk}. The problem of Blocker is still straightforward. To prevent Runner, who is at v_j, from winning she needs to destroy the total mass (capacity) of $\sum_k w_{jk}$ of edges while she has c cuts at her disposal. The vertex v_j is then labeled with the number $L(v_j) = -c + \sum_k w_{jk}$, which is just the number of extra cuts Blocker needs to win.

Now, moving up the tree towards the root, consider the vertices whose all children has already been labeled. Take any such vertex, say v_i, it has children indexed by j and the weights of corresponding edges are w_{ij}. Here, Blocker faces the key trade-off allowed by global cutting rules. To prevent a win by Runner along the path from v_i via v_j she needs either w_{ij} cuts or L_j cuts. Optimal behaviour of Blocker dictates that she should block a path by making as few of the cuts as possible, that is

$$min\{w_{ij}, L(v_j)\}.$$

The total number of cuts needed to prevent Runner at v_i from winning is, then, just the sum of the cuts along all the paths that start at v_j:

$$\sum_j min\{w_{ij}, L(v_j)\}.$$

The summation is over all children of vertex v_i. The number

$$L(v_i) = -c + \sum_j min\{w_{ij}, L_j\},$$

which is the number extra cuts Blocker needs to win, is the value of the labeling function at v_i.

Note that by construction, L_i takes into account that the cuts are made in the optimal manner at all v_j's, as described by L_j's. Continuing in this manner, the

whole tree can be labeled completely. If the root is labeled with positive number, by construction it implies that Blocker does not have enough overall cut capacity to prevent Runner from winning. Otherwise, if the root's label is non-positive then Blocker is the winner. The only drawback of this labeling approach is that while determining who is to win the game, it does not at the same time necessarily deliver the optimal strategy for Blocker.

The labeling procedure (and the associated labeling function) for sabotage games on the weighted trees with Blocker with cut capacity c constrained by local cutting rule is the defined as:

1. Labeling function $L : V \to \mathbb{Z} \cup \infty$.

2. Label all terminal vertices with "∞".

3. For all vertices with children labeled at the previous step, label a vertex v_i with $(-c + \sum_j \min\{w_{ij}, L_j\})$ where the sum is over all children of that vertex and w_{ij} is the weight of the corresponding edge.

4. Repeat

Thus, we proved the following main result:

Proposition 5. *Suppose G is a weighted tree and Blocker's cut capacity is c under global cutting rule. Runner wins if and only if the starting vertex is labeled with a strictly positive number.*

5 Weighted graphs

5.1 Transforming graphs into trees

We start with an observation that the set of all possible paths between any two distinct vertices on a connected weighted graph $G = (V, w)$ can be represented by a weighted tree. The following procedure that associates a tree to the graph, called tree unraveling of the graph, is straightforward but its formalization requires some additional notation.

Given the weighted graph, denote the set of all paths between the starting vertex and the destination vertex by P. A subpath of the given path

$$(v_1, \ldots, v_n)$$

is a sequence of vertices

$$(v_1, \ldots, v_m),$$

where $1 \leq m < n$. Denote the set of all possible subpaths of the paths in P by V^P. This will be the vertex set of the associated tree. The associated tree has an edge between two u and u' in the associated tree if and only if $u = (v_0, \ldots, v_i)$ and $u' = (v_0, \ldots, v_i, v_j)$. The weight of this edge, $w^P(uu')$ is equal to the weight of the edge $v_i v_j$ in the original graph. Since, by definition, a path is a sequence of distinct

vertices (thus it does not contain cycles), the resulting graph $G^P = (V^P, w^P)$ is indeed a weighted tree.

To illustrate how the procedure works, consider the example in Figure 3. On this graph there are four possible paths from a to d, namely abd, $abcd$, acd, and $acbd$. Thus, the set of vertices of the associated tree is

$$\{v_0 = a, v_1 = ab, v_2 = ac, v_3 = abd, v_4 = acd,$$
$$v_5 = abc, v_6 = acb, v_7 = abcd, v_8 = acbd\}.$$

And the set of the edges of the associated tree is

$$\{v_0v_1, v_0v_2, v_1v_3, v_1v_5, v_2v_6, v_2v_4, v_5v_7, v_6v_8\}.$$

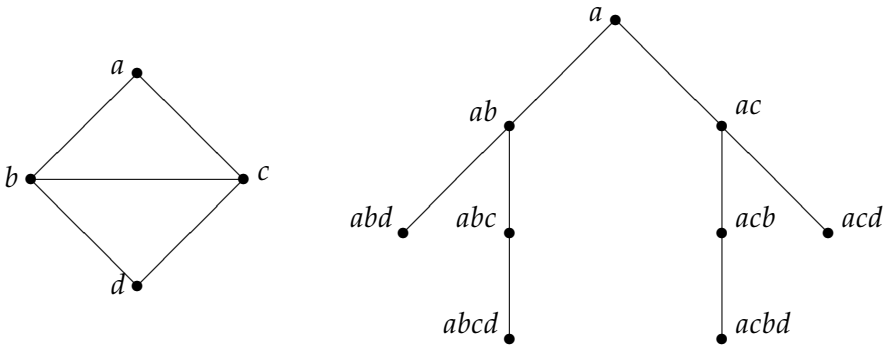


Figure 3: A graph and its associated tree.

Instead of solving the games where there are multiple paths to a single destination, the transformation of the graph into the tree allows us to deal with the games where there are multiple destinations with the unique path leading to each. This transformation can be viewed as a simplification of the original graph that by getting rid of the cycles 'straightens' all the paths to destinations.

At the same time the transformation preserves (in case of a local Blocker) all the relevant strategic information. For every vertex $v_0 \dots v_k$ of the associated tree the set of edges incident with this vertex is the subset of the edges incident with the vertex v_k of the original tree. If an edge incident with v_k in the original graph is absent in the associated tree then, by construction, it implies that any path from v_k via this edge to the destination would contain a cycle.

However, as was argued earlier, any optimal strategy of Runner on a tree can be formulated as the optimal strategy without cycles. Thus, excluding those edges does not affect the strategic opportunities available to Runner. It leads to the following important result:

Theorem 1. *Suppose G is a weighted graph and Blocker's cut capacity is c under the local*

cutting rules. Runner wins on G if and only if Runner wins on the associated weighted tree G^P.

Proof. Suppose Runner wins on the original graph. Since Blocker obeys local cutting rules, whenever it is her turn to move, her move on the associated tree involves deleting edges incident with the current vertex of Runner, $v_0 \dots v_k$. To any edge incident with this vertex there corresponds the edge incident with v_k in the original graph. Thus, to every move of Blocker on the associated tree there corresponds the move of Blocker on the original graph. Since Runner has a winning strategy on the original graph, Runner has a winning reply to every move of Blocker at v_k. And, since Runner's optimal strategy does not involve cycles, to this winning move of Runner at the vertex v_k there corresponds a winning move of Runner at the vertex $v_0 \dots v_k$ on the associated tree.

Suppose, conversely, that Runner wins on the associated tree. By Lemma 1, any optimal strategy of Blocker can be stated as local optimal strategy. Thus, Runner on the tree has optimal reply to any move of Blocker. And Runner is always able to replicate this winning reply on the graph, thus guaranteeing himself a win. □

Under global blocking rules, the result is less general. If Runner wins on the original graph then Runner still wins on the associated tree. The converse is no longer true, however, as example in Figure 4 illustrates. The intuition behind it is that the transformation of the original game into the game on the associated tree restricts the actions available to Blocker in the way that makes it more difficult for her to win.

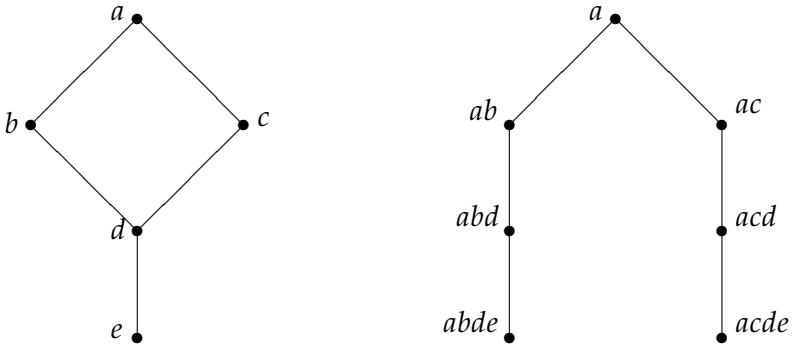


Figure 4: A weighted graph with weights $w(ab) = w(ac) = 1$, $w(bd) = w(cd) = 2$, $w(de) = 3$ and its associated tree. Blocker wins on the graph but not on the associated tree.

Theorem 2. *Suppose G is a weighted graph and Blocker's cut capacity is c under the global cutting rules. If Runner wins on G then Runner wins on the associated weighted tree* G^P.

Proof. Observe that the sabotage game is a finite strictly competitive game that cannot end in a draw. In other words, the sabotage games has exactly two outcomes: either

Runner wins or Blocker wins. As a result, the theorem can be restated in the equivalent (contrapositive) form: If Blocker wins on G^P then Blocker wins on G.

Suppose that Blocker wins on the associated tree G^P. Since Runner obeys local moving rules, whenever Runner is able to move, his move on the graph involves travelling from his current vertex to an adjacent vertex v_k. To this vertex there corresponds vertex $v_0 \dots v_k$ of the associated tree. Since Blocker has a winning strategy on the associated tree, Blocker also has a winning reply to every move of Runner on the original graph. □

6 Application: Border Security

The Mexico-United States border is one of the world's largest drug smuggling corridors. Seventy percent of all cocaine, up to 80 percent of all marijuana and 30 percent of the heroin entering the United States comes from Mexico (see Pulat (2005)).

The assets employed by the United States Border Patrol (USBP) to interdict illegal entries (infiltration) can be broadly divided into two categories: the detection assets and the interception assets. The major detection assets are sensors (small seismic or magnetic transmitters that are capable of detecting movement on the ground and transmitting the information to the nearest border patrol station) and heli-patrolls (helicopters flying at low altitiudes). Much lesser role is played by fences, physical barriers, and different types of lighting. The interception (or capture) is performed by road and off-road patrols.

The set of all possible infiltration routes through the given border area (determined by the existing roads, paths, and physical characteristics of the terrain) constitutes the infiltration network. The structure of the network is common knowledge both to the infiltrator and the interdictor.

The infiltrator's objective is to avoid getting caught while traveling from the southern side of the border (a starting vertex of the network) to north of the area of responsibility of USBP's stations in the region (the set of termination vertices). The objective of the interdictor is to prevent the infiltration. The interdictor decides how to allocate limited defensive resources in anticipation of infiltration. More specifically, the interdictor decides on the location of its detection assets and the location of the border patrol stations.

Washburn and Wood (1995) pioneered the game-theoretic approach to the class of network interdiction problems by modeling it as a two-person zero-sum game. They consider a problem where an infiltrator aims to travel from a specified source to a specified destination while an interdictor tries to detect the infiltrator by setting up inspection points on some of the edges. Each edge has a known probability of detection and interception if the infiltrator decides to use that edge while the interdictor has an inspection point there. The solution of such game is typically a mixed "path selection" strategy for the infiltrator and a mixed "edge-inspection"

strategy for the interdictor.

Several important real-life features of the border security problem cannot, however, be addressed within Washburn and Wood framework. First, detection and interception of an infiltrator are not treated separately. As mentioned above, the detection of possible infiltration is mostly performed by the remote sensors, after which the nearest border station can direct the border patrol to the specified part of the network. Second, traversing the network is inherently a dynamic process. The infiltrator who was detected once but avoided the capture, can later on be detected by the different detection asset and the another attempt at capture can be made in the different part of the network. Third, most of the interdictor's assets are transparent, their locations are perfectly observed by the infiltrator.

Without developing a complete model, we present framework (based on sabotage games developed in the previous section) that allows to model and address all three of these issues.

Consider a sabotage game. The infiltrator plays the role of the runner and the interdictor plays the role of the blocker. Without loss of generality, the detection assets are identified with the nodes of the network and the edges correspond to the paths, roads, or areas that can be patrolled. The destruction of an edge is interpreted as the deployment of the patrol in the corresponding path, road, or area.

It is reasonable to assume that interdictor obeys local cutting rules. While border patrols are reasonably mobile within their area of responsibility, their transportation to the different areas might be costly. The more restrictive assumption is that the cut capacity of interdictor is constant for every node. This is equivalent to saying that all border station have the same capability of capture. The heterogeneity of the border stations can be taken into account by modifying the labeling function.

The Theorem 1 can then be immediately put to use to predict the ability of interdictor to prevent infiltration in a given network. As the next step, a higher-order optimization problem can be solved: whether it is possible, in a given network, to reallocate the available detection and capture assets to prevent the infiltration or minimize the number of infiltration routes.

This framework can be further extended to include an important real-life feature of the border security problem: stochastic nature of interception process. Even if a patrol or a road block is activated along a possible infiltration route, the success of interception is not necessarily assured. A slight modification of the rules of the game allows to model it as well.

Suppose that if infiltrator chooses to travel trough a blocked edge he is able to traverse it with some exogenously given probability ρ. The probability $1 - \rho$ is interpreted as the effectiveness of the interceptor. Suppose that the probabilities of interception on blocked edges are independent of the identity of a blocked edge and of each other. Then, the total number of the blocked edges the infiltrator travels on a given route from origin to a destination completely determines the probability of

successful infiltration on that route. Note that under such assumptions the game is still zero-sum.

A modified labeling function, that counts the number of blocked edges the infiltrator must travel through is constructed in a manner similar to one used in proofs of Propositions 2-5.

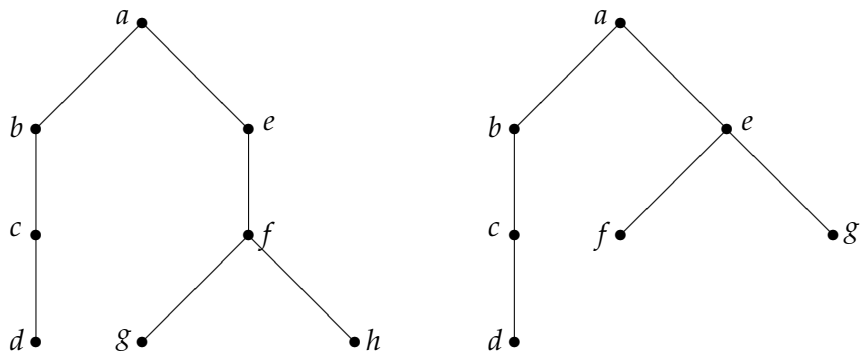


Figure 5: A stochastic border control problem.

As an illustration, consider border control networks in Figure 5. In the first network (on the left), there are two subgame perfect equilibria and in both of them the maximal number of blocked edges that the infiltrator can be forced to travel through is two. In both equilibria the interceptor starts by blocking the edge ae. If the infiltrator responds by going to b via the non-blocked edge ab, then the interceptor would block the edge bc and, if needed, the edge cd later. If the infiltrator tries to go to e via the blocked edge ae and succeeds in avoiding the capture, then the interceptor would optimally respond by blocking the edge ef. In the second network (on the right), there is a unique subgame perfect equilibrium in which the maximal number of blocked edges that the infiltrator can be forced to traverse is one. The interceptor again starts by blocking the edge ae. Now, however, it is suboptimal for the infiltrator to avoid the blocked edge since the infiltrator would then face two blocked edges (bc and cd).

7 Conclusions

This paper has introduced and studied several generalizations of the original sabotage game and characterizes equilibrium strategies for several sets of rules.

There are several further research questions that are promising. The first one is to consider games in which Runner has access to global moving rules. The example in Figure 1 treats players asymmetrically: while Blocker enjoys global cutting rule, movements of Runner are always local. Having both player to obey global rules leads to an easy win by Runner. If Blocker starts by deleting the edge bs or one of the edges between a and b, Runner wins by securing one of the edges between a and l or between l and s. No matter what Blocker does, Runner wins on the next move by

completing the path between s and a via l. Otherwise, for any other opening move of Blocker, Runner wins by securing the edge bs. No matter what Blocker does, Runner wins on the next move by securing an edge between a and b. The sabotage games in which both players have access to global rules would encompass the entire class of so-called connection games (Browne, 2005) from Tic-tac-toe (or Noughts and Crosses) to Hex, TwixT, etc.

The second open research questions is the connection to Ford and Fulkerson (1956) and related results. The max-flow min-cut theorem can be interpreted as an answer to the following question: what is the minimal cut capacity needed by Blocker to win if she is allowed only one cut that is determined at the beginning. Sabotage games pose a dynamic version of the question: what is the minimal cut capacity needed by Blocker if the cuts are made sequentially and are subject to the movement of Runner on the graph.

References

[1] Browne, C., Connection Games: Variations on a Theme. AK Peters, Massachusetts (2005).

[2] Berlekamp, E.R., Conway, J.H., Guy, R.K., Winning Ways for Your Mathematical Plays, Volume 2: Games in Particular. Academic Press (1982).

[3] Collado, R.A., Papp, D., Network interdiction - models, applications, unexplored directions. RUTCOR research report, no. 4-2012. (2012).

[4] Dell, M., Trafficking networks and the Mexican Drug War. MIT working paper (2014).

[5] Ford, L.R., Fulkerson, D.R., Maximal flow through a network. Canadian J. Math. 8, 399-404 (1956).

[6] Gerchick, M., Full Upright and Locked Position. W. W. Norton & Co (2013).

[7] Hong, S., Strategic network interdiction. Fondazione Eni Enrico Mattei working paper 43-2011 (2011).

[8] Jackson, M.O., Social and Economic Networks. Princeton University Press (2010).

[9] Jackson, M.O., Zenou, Y., Games on networks. In: Handbook of Game Theory, Volume 4. Elsevier Science (2015).

[10] Klein D., Radmacher F.G,, Thomas, W., Moving in a network under random failures: A complexity analysis. Sci. Comput. Program. 77, 7-8, 940-954 (2012).

[11] Löding, C., Rohde, P., Solving the sabotage game is PSPACE-hard. In: Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science, MFCS 2003, Lect. Notes Comput. Sc. 2747, 531-540, Springer (2003).

[12] McBride, M., Hewitt, D., The enemy you canŠt see: An investigation of the disruption of dark networks. J. Econ. Behav. Organ. 93, 32-50 (2013).

[13] Nowakowski, R., Winkler P., Vertex-to-vertex pursuit in a graph. Discrete Math. 43, 2, 235-239 (1983).

[14] Papadimitriou, C.H., Games against nature. J Comput. Syst. Sci. 31, 2, 288-301 (1985).

[15] Radmacher, F.G., Thomas, W., A game theoretic approach to the analysis of dynamic networks. Electron. Notes Theor. Comput. Sci. 200, 2, 21-37 (2008).

[16] Rohde, P., Moving in a crumbling network: The balanced case. In: Proceedings of the 18th International Workshop on Computer Science Logic, CSL 2004, Lect. Notes Comput. Sc. 3210, 310-324, Springer (2004).

[17] van Benthem, J., An essay on sabotage and obstruction. In: Mechanizing Mathematical Reasoning, Lect. Notes Comput. Sc. 2605, 268-276, Springer (2005).

[18] Washburn, A., Wood K., Two-Person zero-sum games for network interdiction, Oper. Res. 43, 2, 243-251 (1995).