On Sabotage Games

Dmitriy Kvasov\textsuperscript{1}

\textsuperscript{1}The University of Adelaide

ANU, 27 March 2014
Introduction: I

Figure: A sabotage graph.

[Source: van Benthem (2000)]
The cutting rule specifies the actions available to Demon as the game progresses.

1. We say that the Demon obeys local rule if the edges the Demon may delete are those incident with the vertex the Logician is currently at.

2. The global rule allows Demon to delete an edge anywhere in the graph, independent of the current position of the Logician.

3. Note: On a graph, with local cutting rule and the single destination, solving the game is straightforward: Demon lays in wait doing nothing till the Logician reaches a vertex adjacent to the destination. Then, the Demon cuts the edge in question. The solution is not as obvious if the set of destinations has more than one vertex.
The example game treats players asymmetrically. Demon enjoys global cutting rule, but movements of the Logician are always local: he only goes from a vertex to the adjacent one.

A global movement rule for the Logician would allow him to “secure” an edge anywhere in the graph and once secured, the edge cannot be cut by the Demon. The Logician wins if he is able to secure a path from the origin to a destination.

Having both player to obey global rules makes the game truly symmetric. In example, Logician has an easy win (note that of course it doesn’t matter where the game starts) by securing edge between $B$ and $S$. No matter what the Devil does, Logician wins on the next move by securing an edge between $A$ and $B$. 
We allow the Demon to cut more than one edge at a time, or, more precisely, Demon is characterized by a cut-capacity, that is the total capacity of the edges she is allowed to cut at any move.

Suppose the Demon has cut-capacity $c = 2$. Logician starts by securing $AL$, the Demon needs to cut $LS$ right now, after which the Logician secures $BS$, threatening to win with $BA$ or $BL$ on the next move. To block both the threats,Demon needs to cut the links of the total capacity of 3, which is beyond his power, so Logician wins.
**Definitions: I**

1. A graph $G$ is a finite non-empty set $V$ of vertices and a set $E$ of edges (unordered pairs of distinct vertices) that connect some of the vertices. Two vertices joined by the edge are called adjacent.
2. A path on the given graph is a sequence of vertices, $v_1, v_2, \ldots, v_n$ s.t. all vertices are distinct (except possibly, $v_1$ and $v_n$) and any two consecutive vertices are adjacent.
3. A cycle is a path with three or more vertices in which the first and the last vertices are the same ($v_1 = v_n$).
4. A graph is connected if there is a path between each pair of vertices.
A **weighted graph** is a graph $G$ and a weight function $w$ that associates a positive integer (weight or capacity) with every edge in the graph,

$$w : E \to \mathbb{Z}.$$

A graph is a weighted graph with all the weights equal to one.
“A fool sees not the same tree that a wise man sees.”

— William Blake, "The Marriage of Heaven and Hell”, 1790-3

Instead of looking at games where there are multiple paths to a single destination, we look at the games where there are multiple destinations with the unique path leading to each.
A **tree** is a connected graph with no cycles.

A **rooted tree** has a distinguished vertex called the root, it induces natural partial order on edges, towards (or away) from the root.

A **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 \( T \).
A two-player sabotage game:

1. a connected graph $G$,
2. a starting vertex $s \in V$,
3. a non-empty set of destinations, $D \subset V$, $s \notin D$.
   Note: the set of destinations may contain more than one vertex.
4. cutting rule that specifies actions available to Player 1 (Demon),
5. movement rule that specifies actions available to Player 2 (Logician),
6. payoff function.
Players move sequentially starting with Demon.

At each move Demon can delete edge(s) subject to the restriction that the total weight of the deleted edges does not exceed Demon’s cut-capacity, $c \in \mathbb{Z}$, $c$ is strictly positive.

Logician starts at $s$ and can travel to any vertex adjacent to the one he is currently at. (local rule)

Game ends when Logician reaches a vertex with no children. If that vertex is in $D$, then Logician wins; otherwise Demon wins.
To find Demon’s optimal strategy on trees it suffices to consider strategies that utilize only the local cutting rules.

**Lemma 1**

On simple trees, any optimal strategy of Demon can be stated as local optimal strategy, i.e. the strategy that involves deleting the edges incident to the current vertex of Logician.

**Proof.**
The games on simple trees such that every non-terminal node has at most two children admit a very straightforward and intuitive characterization.

**Theorem 2**

*Suppose that every non-terminal vertex has at most two children. Logician wins if and only if every non-terminal vertex has exactly two children.*

**Proof.**
An intuitive way to extend the previous proposition to the case of arbitrary trees of unitary capacity utilizes the notion of the induced subgraph.

- A graph $G' = (V', E')$ is a subgraph of $G = (V, E)$ if $V' \subset V$ and $E' \subset E$, written $G' \subset G$.
- If $G'$ contains all edges of $G$ that join two vertices in $V'$ then $G'$ is said to be the subgraph induced by $V'$. 
Theorem 3

On a tree $G$ Logician wins if and only if there exists an induced subgraph $G'$ of $G$ such that every non-terminal vertex of $G'$ has exactly two children.
Instead of proving the proposition directly, we introduce a “labeling” procedure (thinly disguised backward induction procedure). The labeling approach then used to prove this and the next results for weighted trees.

Labeling procedure:
- Label all terminal vertices $W$
- For all vertices with children labeled at the previous step, label a vertex $W$ if it has more than one $W$-child, otherwise label it $L$
- Repeat
Theorem 4

*On a simple tree* $G$, Logicians wins if and only if the root is $W$-vertex.

*Proof.*
Local and global cutting rules are NOT equivalent anymore!
The main idea of lemma 1 is that it’s equally difficult to cut a path at any edge, so that can as well be done as early as possible.
In weighted trees a path can have a “weak link”, moreover, it can happen for two different reasons.

1. First, an edge is weak if it has lower weight than the other edges in the path.
2. Second, there as a sort of “depreciation” of the edges’ weights along the path. Sometimes, Demon can prevent Logician from arriving at his destination because Demon can destroy that edge by the time it takes Logician to reach it. Exactly for this very reason Demon is able to beat Logician in the game in the Introduction when Logician is going from $A$ to $S$. 
Weighted trees: II

Labeling procedure:

1. Label all terminal vertices $W$.
2. For all vertices with children labeled at the previous step, label a vertex $W$ if sum of the weights of edges incident to $W$-child is more than $c$, otherwise label it $L$.
3. Repeat.

Theorem 5

Suppose $G$ is a weighted tree with local cutting rules and Demon’s cut capacity is $c$. Logicians wins if and only if the starting vertex is labeled $W$. 
The main idea: to use recursion to determine—at every vertex—the minimal optimal number of cuts for Demon. That number is the value of the labeling function $L$.

All terminal vertices are labeled with $\infty$.

All vertices $v_j$ whose only children are terminal vertices labeled with $L(v_j) = -c + \sum_k w_{jk}$, which is the number of extra cuts Demon needs to win.

All vertices $v_i$ with children already labeled. Demon faces the key trade-off: to prevent a win along the path from $v_i$ via $v_j$ she needs either $w_{ij}$ cuts or $L_j$ cuts. Optimality dictates: $\min\{w_{ij}, L(v_j)\}$. The total number of cuts needed is the sum of the cuts along all the paths that start at $v_j$: $\sum_j \min\{w_{ij}, L(v_j)\}$.

Drawback: while determining who is to win the game, it does not at the same time deliver the optimal strategy for either player.
Weighted trees: IV

- Labeling function \( L \rightarrow \mathbb{N} \).
- Label all terminal vertices with “0”.
- For all vertices with children labeled at the previous step, label a vertex \(-c + \sum \min \{w_i, (L_i - c)_+\}\) where the sum is over all children of that vertex and \(w_i\) is the weight of the corresponding edge.
- Repeat

**Theorem 6**

Suppose \( G \) is a weighted tree with local cutting rules and Demon’s cut capacity is \( c \). Logician wins if and only if the starting vertex has label zero.
Transformation

- Convert $G$ to the associated weighted tree $G^P$. 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. Provides sufficient condition.

Theorem 7

Suppose $G$ is a weighted tree and Demon’s cut capacity is $c$ under the global cutting rules. If Logician wins on $G$ then Logician wins on the associated weighted tree $G^P$. 
Global movement rules (for Logician)
Example:
The multigraph in the Introduction can be represented by the weighted graph with the vertices

\[ V = \{ A, B, L, K, S \}, \]

the edges

\[ E = \{ AB, AL, AK, BL, LK, BS, LS, KS \}, \]

and the weights

\[ w(AB) = w(AL) = w(LS) = 2, \]
\[ w(AK) = w(BL) = w(LK) = w(BS) = w(KS) = 1. \]
Proof of lemma.
Suppose that at some moment in the game, Player 2 is at the vertex $v_t$ but the optimal strategy of Player 1 requires deletion of an edge not incident with the current position of Player 2, say, deletion of an edge at the vertex $v_s$.
Consider a strategy of Player 1 that instead removes an edge incident with $v_t$ and on the path to $v_s$.
Thus modified strategy of Player 1 is clearly also optimal as it reduces by one the number of moves available to Player 2 at $v_t$ and coincides with the original one for all moves now available to Player 2 at $v_t$.
This procedure can be repeated for all non-local moves of Player 1, reducing a global strategy of Player 1 to a local one in the finite number of steps.
Proof of Theorem 2:
Suppose there is a non-terminal vertex with only one child, say node $v_k$. Consider a path from $v_0$ to $v_k$: Note that since $G$ is a tree, this path is unique. D can force L to follow that path, thus winning the game. At his $i$'s move ($i = 0, 1, \ldots, k - 1$), D removes the edge at $v_i$ that is not incident to $v_{k+1}$. If there is no such edge, D removes any other edge or does nothing. Thus at every move, L is forced to move along the path from $v_0$ to $v_k$.
If every vertex has exactly two children, then L has a winning strategy, because after any move of D, L can move to the vertex such that all non-terminal children of that vertex have exactly two children.
Proof of Theorem 4:
Given a labeled tree with the root labeled $\mathcal{W}$ it is easy to construct a winning strategy for Player 2, as all hard work is already done while labeling. All Player 2 needs to do at every move is to go from a $\mathcal{W}$-vertex to another $\mathcal{W}$-vertex. Player 1 cannot hinder him in that, as any $\mathcal{W}$-parent vertex has at least two $\mathcal{W}$-children vertices.

Suppose, conversely, that Player 2 has winning strategy, then it must be the case that any vertex that Player 2 traverses has at least two children. Otherwise, the path through that vertex would have been blocked by Player 1. Thus, the labeling procedure would assign $\mathcal{W}$ to every non-terminal vertex on that path, including the starting one.