Analyzing Network Topology for DDoS Mitigation Using the Abelian Sandpile Model*

Bhavana Panchumarthi† and Monroe A. Stephenson‡

Abstract. Using the Abelian Sandpile model, we have created a new toy model that can optimize the DDoS mitigation strategy of blackholing. The model captures the phenomenon of how much data is lost given a distinct server that is DDoSed, taking into account internal deletion and external rejection. Though useful for DDoS mitigation, the underlying mathematics that the model presents are also deeply profound. In particular, the properties of the matrices constructed to represent every potential combination of black hole and attacked servers are worth exploring algebraically.

Key words. Abelian Sandpile Model, Network Protocols, Stochastic and Deterministic Network Models

AMS subject classifications. 05C25, 68M12, 90B10, 90B15

1. Introduction.

A Distributed Denial of Service (DDoS) is a cyber attack, which is capable of triggering a cascading failure in the victim network. While DDoS attacks come in different forms, their general goal is to make a network's service unavailable to its users. This paper will be considering a SYN Flood, which is a form of DDoS attack that takes place when an "army" of distributed sources overwhelms the targeted server of a network with a large volume of fraud requests, thus rendering it unable to respond to legitimate traffic [2]. With the victim server overloaded, load balancing technology redirects traffic to neighboring servers, potentially overloading them as well. While only a single server might be targeted by the attack, a large part, or even the entirety, of the network thus experiences the attack.

Current optimal solutions to a SYN Flood include frequency-based detection of an incoming attack by monitoring live traffic flow to prevent the onset of a cascading failure [2]. Through DDoS detection, the network can respond with appropriate mitigation methods. A common, but risky, countermeasure is to *blackhole* or *null route* the *source*, or the attacked destination. When a server becomes a blackhole, or referred to as the *sink* in the paper, the data that is assigned to it "disappears" or gets deleted [3]. This method of mitigation may have serious consequences because data traffic that is routed into the sink consists of both legitimate and fraud traffic. Because legitimate traffic is also affected, blackholing may not prevent the DDoS attack from achieving its goal of disrupting the network's functioning. However, since blackholing is considered a viable countermeasure to a SYN Flood, it presents an invitation for further investigation.

This paper shows how mathematical modeling can propose an alternative blackholing

*Submitted to the editors DATE.

Funding: This work was done as part of the 2020 PSU altREU with P.I. Christof Teuscher of PSU under direct supervision of Art Duval of UTEP

†Reed College (panchbh@reed.edu).

‡Reed College (mostephen@reed.edu, people.reed.edu/~mostephen).

strategy that could improve the efficiency of this countermeasure. Aiming for efficiency in terms of obtaining the least possible data loss (which may include both legitimate and malicious traffic), one could ask the following questions: When is the best situation to introduce a sink? Are non-source servers equally feasible, or perhaps even better choices as a sink? How is traffic already in the network at the time of blackholing affected? How is incoming legitimate traffic requesting service while the sink is active affected?

The mathematical model previously mentioned is the Abelian Sandpile Model (ASM). First introduced by Per Bak, Chao Tang and Kurt Wiesenfeld , the model explained the concept of *self-organized criticality* (SOC) in their 1987 paper [1]. Systems exhibiting SOC possess a stochastic dynamics that drives them to a stable, or *critical* state. A couple of years later, Deepak Dhar gave the model an algebraic and combinatorial characterization, as well as the term "Abelian"[5]. Using the notion of SOC, the ASM can be extended to optimize the efficiency of using a blackholing strategy to drive the network to a stable state.

To investigate the optimization of blackholing, we propose a chip-firing game that aims to determine the best server to blackhole, or *enable as the sink*, for each server acting as a potential source. Additionally, this chip-firing game could suggest a feasible server to enable as the sink for situations where mitigation takes priority over identifying the source.

The undirected graphs of Internet backbone networks like Airtel, Cogent Communications and AGIS acted as a playing field for the proposed chip-firing game. Our model, built in Sage, outputs matrices representing values of data preserved for each combination of source and sink. In addition to ranking servers by their susceptibility as a source and efficiency as a sink, analyzing such matrices serves a greater incentive. For example, aiming to understand how the characteristics of a given graph contribute to the structure of a corresponding matrix could further the versatility of the proposed chip-firing game.

2. Background.

2.1. The Abelian Sandpile Model.

At the heart of the Abelian Sandpile Model there exists the algebraic and combinatorial tool: the discrete divisor. For our purpose, a divisor is an element of free abelian group on a graph, assigning a value to each vertex. On each vertex the values assigned corresponds to a finite number of chips or sandgrains. A vertex is stable if the amount of sand on it is less that its degree; otherwise, the vertex is unstable.

A divisor $D \in \mathrm{Div}(G)$ can be written as the summation $D = \sum_{v \in V} D(v)v$, where V is the set of vertices on graph G. In order to determine the total number of grains on a graphs, or the degree of divisor D, we add up the coefficients $D(v)$. Most of the background is inspired and based off of Corry and Perkinson's book [4].

Interpreting network G as an undirected graph, where sand-grains exist on the nodes and can be displaced across G by toppling, gives us the sandpile model. We can witness the abelian property of the sandpile graph at work after we define what it means to fire or topple.

Definition 2.1 (Sandpile Graph). *A sandpile graph is a triple $G = (V, E, s)$ where (V, E) is a graph and $s \in V$ is the sink.*

The above mentioned *sink* refers to the vertex that we denote to be the destination for any sandgrains to simply disappear. Since the amount of sand on a graph is conserved, without the sink, which can accept an infinite amount of excess sand, we will can never stabilize all vertices.

Definition 2.2 (Sink). *For graph $G = (V, E)$ we define the sink to be a globally accessible vertex $s \in V$, and that it has $outdeg_G(s) = 0$. By globally accessible we mean there is a directed path from each vertex to s. We will denote the non-sink vertices as $\tilde{V} = V \backslash \{s\}$.*

Definition 2.3 (Sandpile Configuration). *A configuration is analogous to a divisor on a graph. A configuration of sand on $G = (V, E, s)$ is a an element of the free abelian group on the non-sink vertices ($\tilde{V} = V \backslash s$):*

$$Config(G) = \mathbb{Z}\tilde{V} = \left\{ \sum_{v \in \tilde{V}} c(v)v | c(v) \in \mathbb{Z} \forall v \right\}.$$

With $c(v)$ denotes the amount of sand on the given vertex v. For a sandpile, we require that $Sandpile(G) = \mathbb{Z}_{\geq 0}\tilde{V}$.

Definition 2.4 (Stable Configuration). *A stable configuration is a configuration with no unstable vertices. For a unstable configuration c, its stable state is denoted as c°*

Definition 2.5 (Degree of Configuration). *Suppose we have a configuration c on a graph G, then the degree of c is,*

$$\sum_{v \in \tilde{V}} c(v) \in \mathbb{Z}.$$

We can think of the degree as the amount of total sand grains or chips on a graph G

To displace sand across vertices on a graph, we can *fire* or *topple* vertices.

Definition 2.6 (Vertex Firing). *Let $G = (V, E, s)$. Firing or toppling $v \in \tilde{V}$ from a configuration c produces a new configuration c' such that,*

$$c' = c - outdegree_G(v)v + \sum_{vw \in E | w \neq s} w.$$

We will denote this process as cc'. Only unstable vertices may be allowed to fire within a legal firing, i.e. vertices such that $outdegree_G(v) \leq c(v)$.

Definition 2.7 (Firing Script). *A firing script is a linear combination of the vertices that are being fired.*

For example, if v_1 is fired twice and v_2 and v_3 are each fired once, then we have the firing script $2v_1 + v_2 + v_3$.

We can enumerate the firing script as such:

Definition 2.8 (Degree of a Firing Script). *If $\sigma : V \mapsto \mathbb{Z}$ is a firing script, the degree of σ is*

$$deg(\sigma) := \sum_{v \in V} \sigma(v)$$

The abelian property of the sandpile graph tells us that the order of vertex firings does not matter, or in other words, the sand addition operation is commutative. The inclusion of the abelian property leads to two important theorems about the ASM.

Theorem 2.9 (Least Action Principle). *Let $c \in Config(G)$ and let $\sigma, \tau \geq 0$ be firing scripts such that σ is a legal firing sequence for c and cc° with c° being stable. Then $\sigma \leq \tau$.*[1]

Theorem 2.10 (Uniqueness of Stabilization). *Let c be a configuration and $\sigma, \sigma' \geq 0$ firing scripts corresponding to legal firing sequences for c. Suppose that $c \xrightarrow{\sigma} \tilde{c}$ and $c \xrightarrow{\sigma'} \tilde{c}'$, with $\tilde{c}$ and $\tilde{c}'$ both stable. Then, $\sigma = \sigma'$ and $\tilde{c} = \tilde{c}'$.*[2]

2.2. Definitions of Terms.

To apply the ASM to cascading failure in networks, we need to define some terms:

1. Backbone network: The internet backbone networks we consider serve as examples of networks that could be prone to DDoS attacks. Each network is an undirected graph without an assigned sink [6].
2. Server: A server in a network is a vertex in the undirected graph. When the network load balances, one or more unstable servers fire to redistribute data requests.
3. Data packet: Each data packet, or a data request from a client, is a sandgrain on the undirected graph. A data packet enters the network by the sand addition operation, but cannot exit the graph, except for entering the sink.
4. Blackholing: Blackholing, or null-routing a server, during DDoS mitigation means that any data requests that enter that server are deleted. So, the blackhole of a network is the sink for our purpose.
5. Source: A SYN Flood attack generally targets and overloads a specific server in a network. Here, we refer to that targeted server as the source vertex.

3. Minimization Problem: When and What to Blackhole.

While blackholing the targeted server serves to fix an ongoing cascading failure in a network, focusing on losing less data while doing so makes the strategy more efficient. We may minimize the data lost during the process by considering other servers as potential sinks and the optimal moment to *enable the sink*.

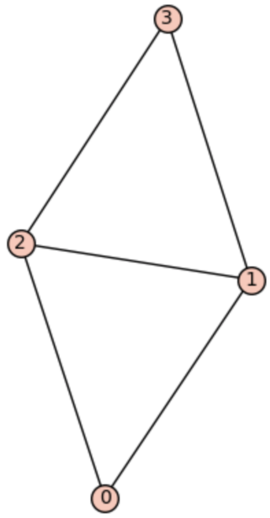
Definition 3.1 (Enabling the Sink). *Suppose we have a graph $G = (V, E)$ with a configuration c_0 on it. As c_0 fires unstable vertices, it becomes different configurations, i.e. $c_0 \xrightarrow{v_0} c_1 \xrightarrow{v_1}$*

[1] Proof in Dave's book.
[2] Proof in Dave's book

. . . . Suppose after firing i vertices we enable the sink at the configuration c_i, so G goes to $\mathcal{G} = (V, E, s)$ with $s \in V$ denoting the enabled sink. We denote this as $G \xrightarrow{s} \mathcal{G}$.

Intuitively, this means that for a vertex s will act as a non-sink vertex for configurations up until c_i. After enabling, the vertex s will act as a sink vertex.

In the Abelian Sandpile Model (ASM), the sink is usually predetermined. Yet, for the analysis of cascading failures we will presume that we start a sandpile without a sink. To explore the problem of when to enable the sink, let us take an example using K_4 minus an edge graph.



Let the sandpile configuration be $c = c_{\max}$. In the case of a SYN Flood on servers, we can overload a single vertex to destabilize the sandpile. The symmetry of the graph allows us to consider only two cases. Either overload a vertex with $\deg_G(v) = 2$ (where $\deg_G(v)$ is the degree of the vertex v on a graph G) or $\deg_G(v) = 3$.

For the first case, we begin with the initial configuration $\{2, 2, 2, 1\}$ (without loss of generality with respect to the other vertex with degree 3). After the initial condition, sandpile configurations can go $\{2, 2, 2, 1\} \rightarrow \{0, 3, 3, 1\} \rightarrow \{2, 1, 1, 3\} \rightarrow \{0, 3, 3, 1\} \rightarrow \{2, 1, 1, 3\} \rightarrow ...$ by legal firings. There are 3 distinct configurations ($\{2, 2, 2, 1\}, \{0, 3, 3, 1\}$ and $\{2, 1, 1, 3\}$) that are potential stages for enabling a sink such that the data loss is minimal. For the $\{0, 3, 3, 1\}$ configuration, the amount of sand lost to each vertex sink is, 2 for v_0 as the sink, 5 for v_1 as the sink, 5 for v_2 as the sink, 3 for v_3 as the sink. We can calculate the other configurations in the cycle to have the same above amounts in their respective sinks.

In the second case we discover the same trend that the total sand in the sink once stabilized is invariant of the configuration in which we enable the sink.

We will now conjecture and prove this generalization.

Theorem 3.2 (Cycle Invariance). *Given a configuration c_i, $(c_i)^\circ$ is recurrent and indepen-*

dent of i.

We must first outline the necessary definitions before proving this theorem.

Definition 3.3 (Recurrent). *A configuration c is recurrent if it is positive ($c \geq 0$), c is stable ($c^\circ = c$), and for a given configuration a, there exists a configuration b geq0, that $c = (a+b)^\circ$*

Definition 3.4 (Reduced Laplacian). *Let $\widetilde{out}(G) = diag(deg(v_1), \ldots, deg(v_n))$ be the diagonal matrix of non-sink vertex degrees and let $\tilde{A}$ be reduced (non-sink) adjacency matrix. Thus, we define the reduced Laplacian as such:*

$$\tilde{L} := \widetilde{out}(G) - \tilde{A}^t$$

Definition 3.5 (Burning Configuration and properties). *We will first define the support of a configuration on G as*

$$supp(c) := \{v \in \tilde{V} | c(v) \neq 0\}$$

and furthermore, the closure of the support, $\overline{supp}(c)$, is the set of non-sink vertices that can be accessed from $supp(c)$ by a path in G that avoids the sink.
Now to define the burning sandpile. A sandpile b on G is a burning sandpile is $b \equiv 0 \pmod{\tilde{\mathcal{L}}}$ and $\overline{supp}(b) = \tilde{V}$.

Proof (Cycle Invariance). We will use induction to prove that $(c_k)^\circ$ is recurrent for all k. For $D_0 = c_{\max} + v_0$ where v_0 is the initial overloaded vertex, we have the following chain,

$$D_0 \xrightarrow{v_0} D_1 \xrightarrow{v_1} \ldots \xrightarrow{v_k} D_k \rightsquigarrow (c_k)^\circ,$$

where each v_k corresponds to an unstable vertex fired so $D_k = c_k + l_k s$, such that $l_k \in$ for some s that will be our chosen sink.

For the base case, we take $k = 0$. By definition of recurrent, $c = (c_{\max} + b)^\circ$ if and only if c is recurrent. Since $(c_0)^\circ = (c_{\max} + v_0)\circ$ takes the same form as the definition of recurrent, $(c_0)^\circ$ must be recurrent.

For the induction hypothesis suppose for some k, $(c_k)^\circ = (c_0)^\circ$. We have two cases:

<u>Case 1</u>: Suppose $v_k \neq s$. Let the firing script for $c_k \to (c_k)^\circ$ be σ. Now let the firing script from $c_k \to c_{k+1}$ be v_k. Finally, let the firing script from $c_{k+1} \to (c_{k+1})^\circ$ be σ'. Thus by the uniqueness of stabilization, since both $(c_k)^\circ$ and $(c_{k+1})^\circ$ are stable, we know that $\sigma = v_k + \sigma'$ and $(c_k)^\circ = (c_{k+1})^\circ$. By transitivity we can see that $(c_0)^\circ = (c_{k+1})^\circ$, and thus $(c_{k+1})^\circ$ is recurrent as well. Diagrammatically we can see this as the commutative diagram below:

$$\begin{tikzcd} c_k \arrow[r, "v_k"] \arrow[dr, "\sigma"'] & c_{k+1} \arrow[d, "\sigma'"] \\ & (c_k)^\circ = (c_{k+1})^\circ \end{tikzcd}$$

Case 2: Suppose $v_k = s$. We can see that firing the sink is equivalent to adding the burning configuration, thus $c_{k+1} = c_k + b$, where b is the burning configuration. By the induction hypothesis, $(c_k)^\circ$ is recurrent. Since we know that $(c_k + b)^\circ = (c_k)^\circ$ (Theorem 7.5), we know $(c_k + b)^\circ$ is recurrent. And since $(c_k + b)^\circ = (c_{k+1})^\circ$ we then have that $(c_{k+1})^\circ$ is recurrent as well.

Now we want to show that all c_k are equal modulo the reduced Laplacian $\tilde{L}$. Once again there are two cases.

Case 1: Suppose $v_k \neq s$. To go from $c_k \xrightarrow{v_k} c_{k+1}$ we have

$$c_{k+1} \equiv c_k - \tilde{L}v_k \equiv c_k \pmod{\tilde{L}}$$

as desired.

Case 2: Suppose $v_k = s$. Note that firing the sink is equivalent to adding the burning sandpile, thus we have

$$c_{k+1} \equiv c_k + b \equiv c_k + \tilde{L}\sigma_b \equiv c_k \pmod{\tilde{L}}$$

where σ_b is the firing script for the burning configuration.

Therefore, we have shown Theorem 3.2, since $(c_k)^\circ$ is recurrent independent of k. ∎

Note, that generalizing this to have an initial configuration that is not $c_{\max}$ is conjectured to be very similar. The general case was not necessary for our purposes, and thus not pursued.

Since the invariance theorem helps us ignore the question of when to enable the sink, we can now consider which vertex to enable as the sink for the least data loss during stabilization. The rest of the paper will focus on this question.

4. Matrix Representation of Sink vs. Source Optimization.

Within the network chip firing, we choose a vertex as a source and a vertex as the sink. We also allow the sink and source to be the same. To determine how much data lost in the sink there is, we must follow the process. First choose which vertex is the source, then choose a sink, then allow for stabilization. Once the sandpile has stabilized, then we count the chips lost to the sink. Since we have shown Theorem 3.2 to be true, we may assume that the sink is enabled from the beginning of the attack.

We can see that we can create a matrix of the amount of data lost on a particular graph. Suppose we have an graph $G = (V, E)$, then the size of the matrix we create would be $V \times V$. Our matrix, called A, expresses the amount of data lost internally within the sandpile. By internally, we suppose that no data is entering or leaving the graph's system. So the only possible data lost is data originating from the initial configuration. Our A matrix would look like

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1V} \\ a_{21} & a_{22} & \dots & a_{2V} \\ \vdots & \vdots & \ddots & \vdots \\ a_{V1} & a_{V2} & \dots & a_{VV} \end{bmatrix},$$

with a_{ij} represents when the $i-th$ vertex is the sink and the $j-th$ vertex is the source.

Definition 4.1 (Entries of the $A-$matrix). *Each entry of the $A-$matrix is*

$$a_{ij} := (c_j + c_{max}) - \sum_{v \in V \setminus v_i} (c_j + c_{max})^{\circ}(v),$$

with $G = (V, E, v_i)$. Here c_j denotes the configuration with one grain on the j-th vertex.

Yet, these are not the only chips the system has lost. We must consider external loss of data, or denied data since the system is down. For every round that the system is not online (there exists an unstable vertex), we consider the rejection of data to be the external loss of data. The amount of data lost in a round is the degree of max stable configuration. This is because we suppose that on each vertex that the net flow of data is 0, with the maximum amount entering and leaving during a given round. When the network is down, no data leaves and no data is accepted into the system, although we can assume there to still be the usual amount of request trying to reach the network.

With the A matrix we ignored the data that should be entering the system, but now we will address that with the B matrix, or the externally lost matrix. First, we must determine how many rounds (or the $\deg(\sigma)$ where σ denotes the legal firing script that brings a configuration to stability) it takes for the system to reach stability. To determine the duration of a network being down, we must decide the sink and the source then proceed to stabilize the sandpile. After determining the amount of rounds it takes, we multiply by the maximum capacity of the sandpile given the enabled sink. We have not yet discovered a method to determine this value besides algorithmically. The matrix is again is a $V \times V$ matrix as such:

$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1V} \\ b_{21} & b_{22} & \dots & b_{2V} \\ \vdots & \vdots & \ddots & \vdots \\ b_{V1} & b_{V2} & \dots & b_{VV} \end{bmatrix},$$

where b_{ij} indicates the external rejection amount for the $i-th$ sink and the $j-th$ source. Again, this is calculated by the amount of rounds it takes for the $i-th$ sink and the $j-th$ source to reach stabilization, multiplied by the max stable configuration for the sandpile with the $i-th$ sink.

To calculate b_{ij} we must first find the stabilization firing script. This can be done using the odometer, Definition 4.2 (Odometer). *odo maps a given configuration to the firing script for its stabilization, i.e.*

$$odo\colon c \mapsto \tilde{L}^{-1}(c - c^{\circ})$$

From here we can easily define b_{ij}.

Definition 4.3 (Entries of B-matrix). *Our rounds of instability is*

$$|\sigma| = |odo(c_j + c_{max})| = |\tilde{L}^{-1}((c_j + c_{max}) - (c_j + c_{max})^{\circ})|$$

From here our b_{ij} entries can be represented in one of 2 ways,

$$b_{ij} = |\sigma| \cdot \deg(D_{max}) \qquad b_{ij} = |\sigma| \cdot \deg(c_{max}),$$

where the latter is a non-constant and depends on i.

From before, we know that we are not only considering single attacks. In order to consider prolonged attacks, we must adjust our definitions above. By the Abelian nature of the sandpiles, it only matters how many times DDoS attacks a given vertex, not in what order. So we may consider that the attacker does all of the attacks first, then we begin to stabilize. Thus the only difference in our definitions is that our initial configuration will be $(n \cdot c_{\mathbf{j}} + c_{\text{max}})$ where n is how many rounds the attacker attacks for. From experimental observations, a prolonged attack only affects the B matrix, not the A matrix. For now, the lack of effect by prolonged attacks on A is conjectured. Computing the A and B matrices is not an easy task without computer-assisted computational methods.

5. Computational Model.

On Sage, we created a model that generates the A and B matrices efficiently, particularly using Perkinson's Thematic Tutorial [9]. To discuss how the model works and how it can help us improve blackholing, we will use the network AGIS and its respective A and B matrices as an example.

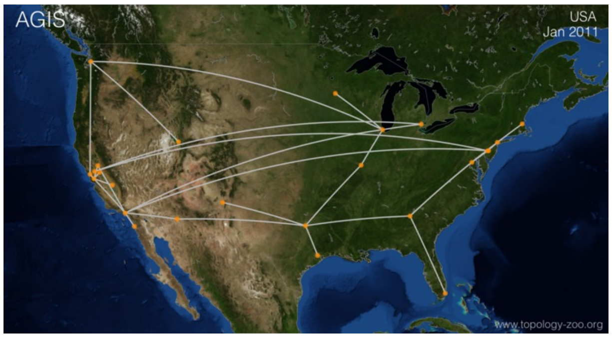


Figure 1. *The Backbone Network AGIS*

The GraphML files for most backbone networks we analyzed, including for AGIS, are from Internet Toplogy Zoo [8]. These graphs act more as a playing field for our model, than as the subject for improvement. Since converting the GraphML files to Python Dictionary was not simple (due to the nature of the files in the dataset), and also to preserve some simplicity in the model, we used weighted undirected graphs. The converted Python dictionaries also do not reflect some information about the bandwith of each edge present in the GraphML files.

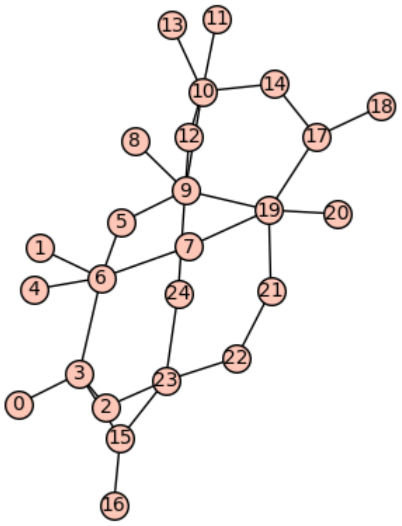


Figure 2. *Sandpile Graph*

In the above the weighted undirected graph of AGIS, the weight of each edge is the number of connections between two servers. Since we are using a labeled graph with 17 nodes (from 0 to 16), we will obtain 17 x 17 A and B matrices as outputs. In both matrices, as discussed earlier, the entries represent the data lost for each combination of source and sink.

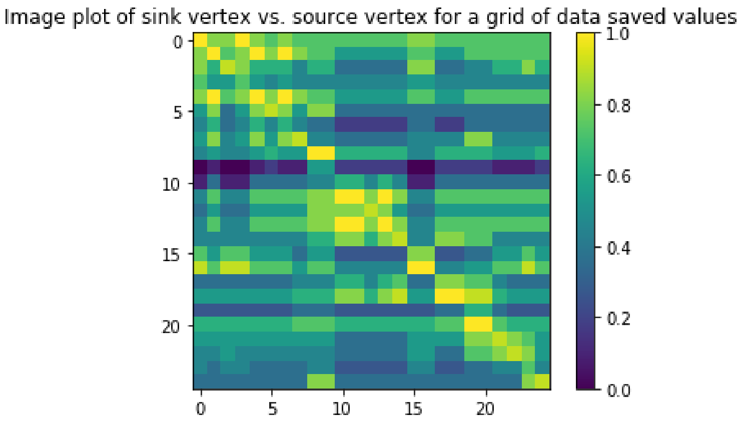


Figure 3. *A Matrix*

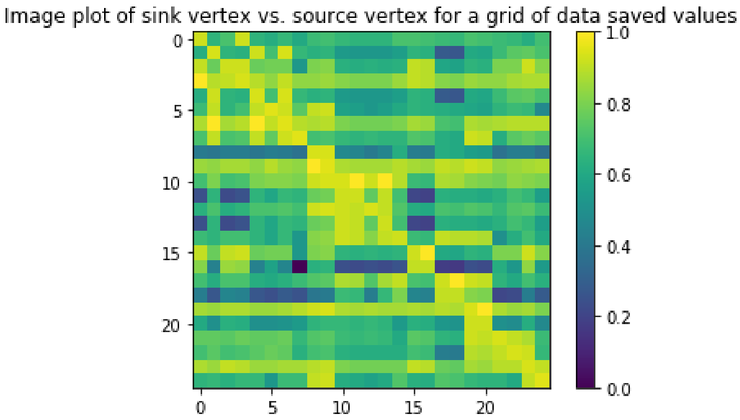


Figure 4. *B Matrix*

The two heatmaps for the normalized A and B matrices help us visualize ideal versus dangerous sink choices for each source vertex. The lighter colors represent the safer combinations, while dark colors represent the more unsafe. We can see that the B matrix heatmap seems to have generally lighter values, while the A matrix heatmap seems to have generally darker and more contrasting values. To obtain a better understanding of the optimal sink-source combinations, we can add the normalized A and B matrices, and obtain the normalized C matrix.

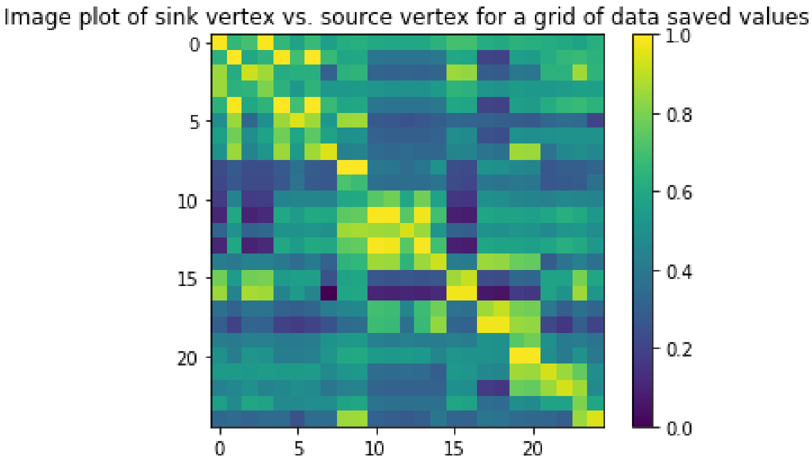


Figure 5. *C Matrix*

The C matrix heatmap for AGIS seems to warn us of quite a few more dangerous combinations than either of the previous two heatmaps.

Thus far in the model, we had been assuming that the DDoS attack does not continue once the network goes past its max stable configuration and when the sink is enabled. So, another parameter we introduced was the length of the DDoS attack. We define this length as the number of rounds of data packet arrival at the source node even after the sink is enabled. The assumption here is that every time a packet arrives, if there are unstable vertices, the network load balances.

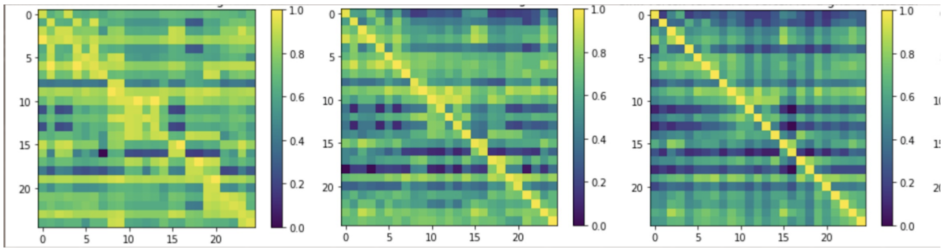


Figure 6. *Prolonged with 1, 5, and 25 rounds*

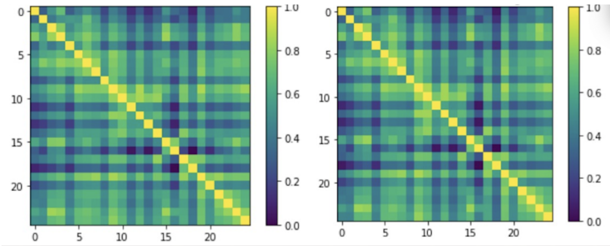


Figure 7. *Prolonged with 100, and 1000 rounds*

While the inclusion of this parameter does not seem to affect the normalized A matrix, we see an interesting trend with the B matrix as the length of the attack approaches infinity. We call this seemingly symmetric matrix we approach the *limit matrix*, which is later included in this paper as Proposition 6.5.

6. Conjectures and Future Directions.

First, we will prove that for the A matrix alone, setting the sink equal to the source optimizes the process.

Proposition 6.1. *For the matrix A, for $i, j \in [0, V]$, $a_{ii} \geq a_{ij}$.*

Proof. If the sink matches the source with $G = (V, E, v_i)$ where $c_\mathbf{j}$ denotes the configuration with one grain on the jth vertex,
then,

$$a_{ii} = \sum_{v \in V \setminus v_i} (c_\mathbf{i} + c_{\max})^\circ(v),$$

and since the extra grain goes directly to the sink,

$$a_{ii} = \sum_{v \in V \setminus v_i} (c_{\max})^\circ(v) = \sum_{v \in V \setminus v_i} (c_{\max})(v) = \deg(c_{\max}).$$

Since $c_{\max}$ is the max amount of chips on G, no other configuration can have more chips. ■

According to the A-matrix, no matter the sink chosen, the source has the least amount of data rejected.

Following is the list of conjectures that appear experimentally true. First, we relate the A matrix with the sandpile group, particularly for the K_n and C_n indicating a more general relation.

Proposition 6.2. *Given a complete graph on n nodes, the sum of all A matrix values is $n-2$ times the sum of all the A matrix values for the cycle graph. This notion is related to the fact that $\mathcal{S}(K_n) \approx \mathbb{Z}_n$ and $\mathcal{S}(C_n) \approx \mathbb{Z}_{n-2}$ where $\mathcal{S}(G)$ is the sandpile group.[7]*

Similarly,

Proposition 6.3. *Given any two trees on n nodes, any column of their A matrices have equivalent sums. This notion is related to the fact that a sandpile group of articulated components is the product of articulated components i.e. $\mathcal{S}(G) \approx \mathcal{S}(G_1) \times \ldots \times \mathcal{S}(G_{n-1})$ since a tree can be decomposed into $n-1$ articulated components. [7]*

When exploring other relations with the A matrix, we experimentally found,

Proposition 6.4. *Suppose we have graphs G and G' and they are cospectral, or having the same characteristic polynomial, then $tr(A_G) = tr(A_{G'})$ and $tr(B_G) = tr(B_{G'})$.*

Finally, both the A and B matrices had effects from prolonged attacks.

Proposition 6.5. *The normalized B matrix has a limit matrix. i.e. a prolonged attack for i rounds with a matrix B_i, with i large enough has the property $B_i = B_{i+1}$.*

Proposition 6.6. *The normalized A matrix is independent of how many rounds of attacks occur. i.e. a prolonged attack for any i rounds with the matrix A_i and a prolonged attack for any j rounds with the matrix A_j has the property $A_i = A_j$.*

All of these conjectures are based solely on experimental observations.

For future research in the area discussed in this paper, there are many places to start. Most immediately is to derive algebraically the A and B matrices so that there is no need to algorithmically compute them. The conjectures above also provide a starting place. Proposition 6.2 and 6.3 are algebraically based involving explicit forms of the Sandpile group that may be expressed in our A matrix. Proposition 6.4 involves exploring the characteristic polynomial of a given graph, and how it relates to the A and B matrices. Finally, Proposition 6.5 and 6.6 might be approached using Markov chains. The A and B matrices have relations to the the statistical and probabilistic nature of the ASM, in particular the burst sizes. We hope that in the future these conjectures and directions are realized to create a more fruitful insight into this particular interpretation of ASM.

7. Appendix A: Algorithms.

Algorithm 7.1 Calculating the A matrix

Require: A graph $G = (V, E)$ that can be a multigraph and directed.
Ensure: A $V \times V$ matrix with the $i - th$ row corresponding to $i - th$ vertex as the sink, and $j - th$ column corresponding to the $j - th$ vertex as the source.
N=0
Full = []
while $N \leq V$ **do**
    P =0
    Rows = []
    S = Sandpile(G, N)
    Max = Max Configuration on S with respect to N sink
    while $P \leq V$ **do**
        One = Configuration of one grain on P source on G
        Final = One + Max
        Stabilize Final
        Deg = Degree (Final)
        Append Deg to Rows
        P = P+1
    end while
    Append Rows to Full
    $N = N + 1$
end while

Algorithm 7.2 Calculating the B matrix

Require: A graph $G = (V, E)$ that can be a multigraph and directed.
Ensure: A $V \times V$ matrix with the $i - th$ row corresponding to $i - th$ vertex as the sink, and $j - th$ column corresponding to the $j - th$ vertex as the source.
N=0
Q= Length of Attack
Full = []
while $N \leq V$ **do**
    P =0
    Rows = []
    S = Sandpile(G, N)
    Max = Max Configuration(S)
    while $P \leq V$ **do**
        One = Configuration of one grain (S)
        Final = Q·One + Max
        I=0
        while Final is not stable **do**
            Fire an unstable vertex on Final
            I=I+1
        end while
        Append I to Rows
        P = P+1
    end while
    Append Rows to Full
    $N = N + 1$
end while

8. Acknowledgements. To be finished at a later point
Art Duval, David Perkinson, Christof Teuscher, Mackenzie Gray, Caroline Klivans, Angélica Osorno, Alison Crocker, our fellow altREU members, and Saketh Panchumarthy.

REFERENCES

[1] P. Bak, C. Tang, and K. Wiesenfeld, *Self-organized criticality: An explanation of the 1/f noise*, Phys. Rev. Lett., 59 (1987), pp. 381–384, https://doi.org/10.1103/PhysRevLett.59.381, https://link.aps.org/doi/10.1103/PhysRevLett.59.381.

[2] E. Chou and R. Groves, *Distributed Denial of Service (DDoS): Practical Detection and Defense*, O'Reilly Media, Inc., 2018.

[3] Cloudflare, *What is blackhole routing?*, Accessed July 2020, https://www.cloudflare.com/learning/ddos/glossary/ddos-blackhole-routing/#:~:text=DDoSblackholerouting/filtering(sometimes,hole, \OT1\textquotedblrightandislost.

[4] S. Corry and D. Perkinson, *Divisors and Sandpiles: An Introduction to Chip-Firing*, American Mathematical Society, 2018.

[5] D. Dhar, *Self-organized critical state of sandpile automaton models*, Physical Review Letters, 64 (1990), p. 1613–1616, https://doi.org/10.1103/physrevlett.64.1613.

[6] W. Hurst, N. Shone, and Q. Monnet, *Predicting the effects of ddos attacks on a network of critical infrastructures*, in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015, pp. 1697–1702, https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.256.

[7] C. J. Klivans, *The Mathematics of Chip-Firing*, CRC Press, Taylor Francis Group., 2018.

[8] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, *The internet topology zoo*, IEEE Journal on Selected Areas in Communications, 29 (2011), pp. 1765–1775, https://doi.org/10.1109/JSAC.2011.111002.

[9] D. Perkinson, *Abelian sandpile model*, https://doc.sagemath.org/html/en/thematic_tutorials/sandpile.html.