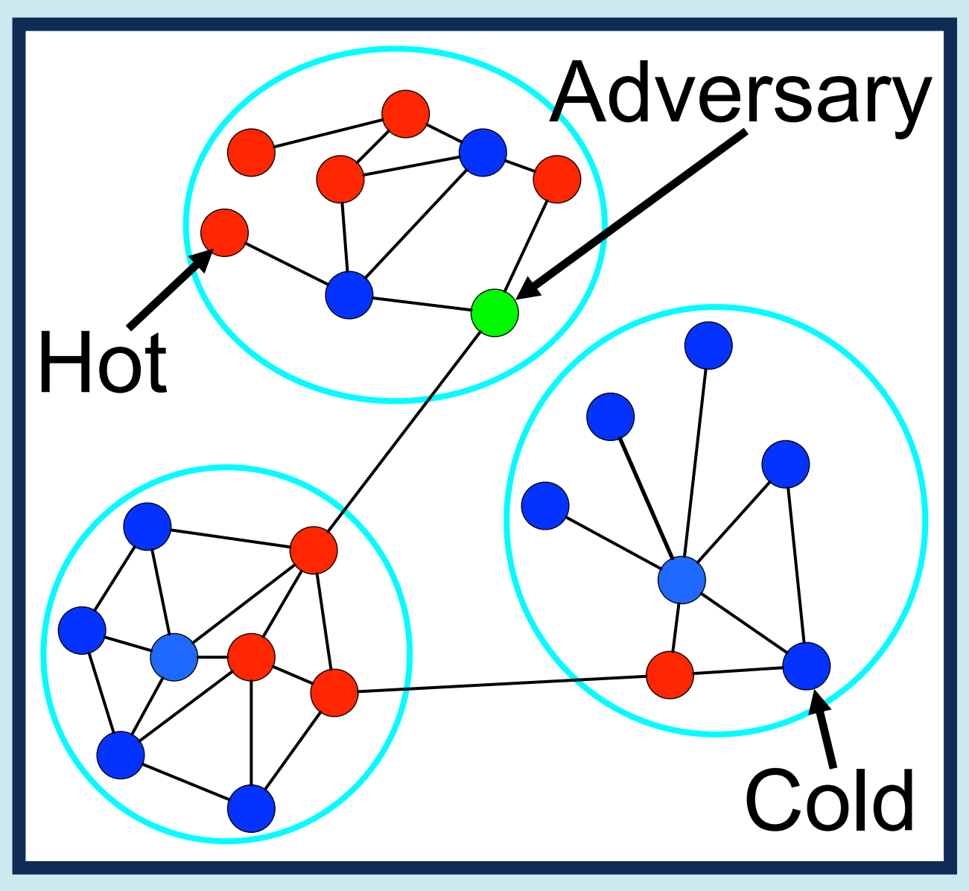
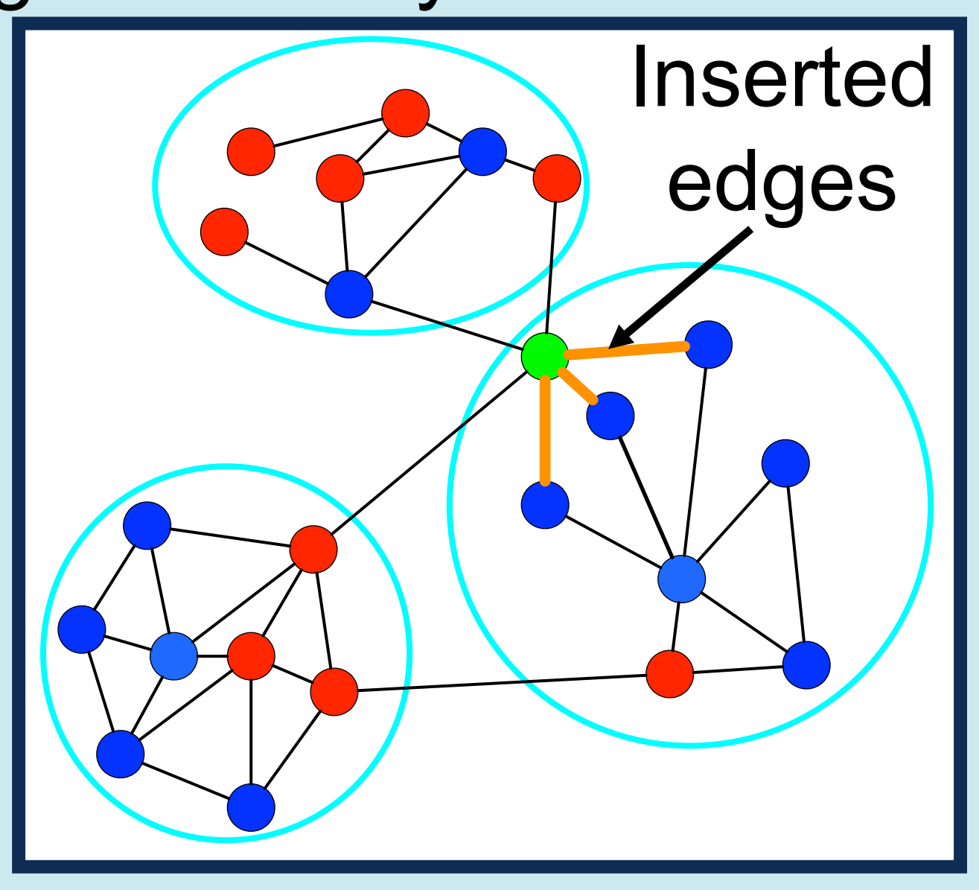
Counter-Adversarial Community Detection

Sandia National Laboratories
Phillip Kegelmeyer, Jeremy D. Wendt, Ali Pinar, Kristen Altenburger
Sandia National Laboratories, California 94551

Problem

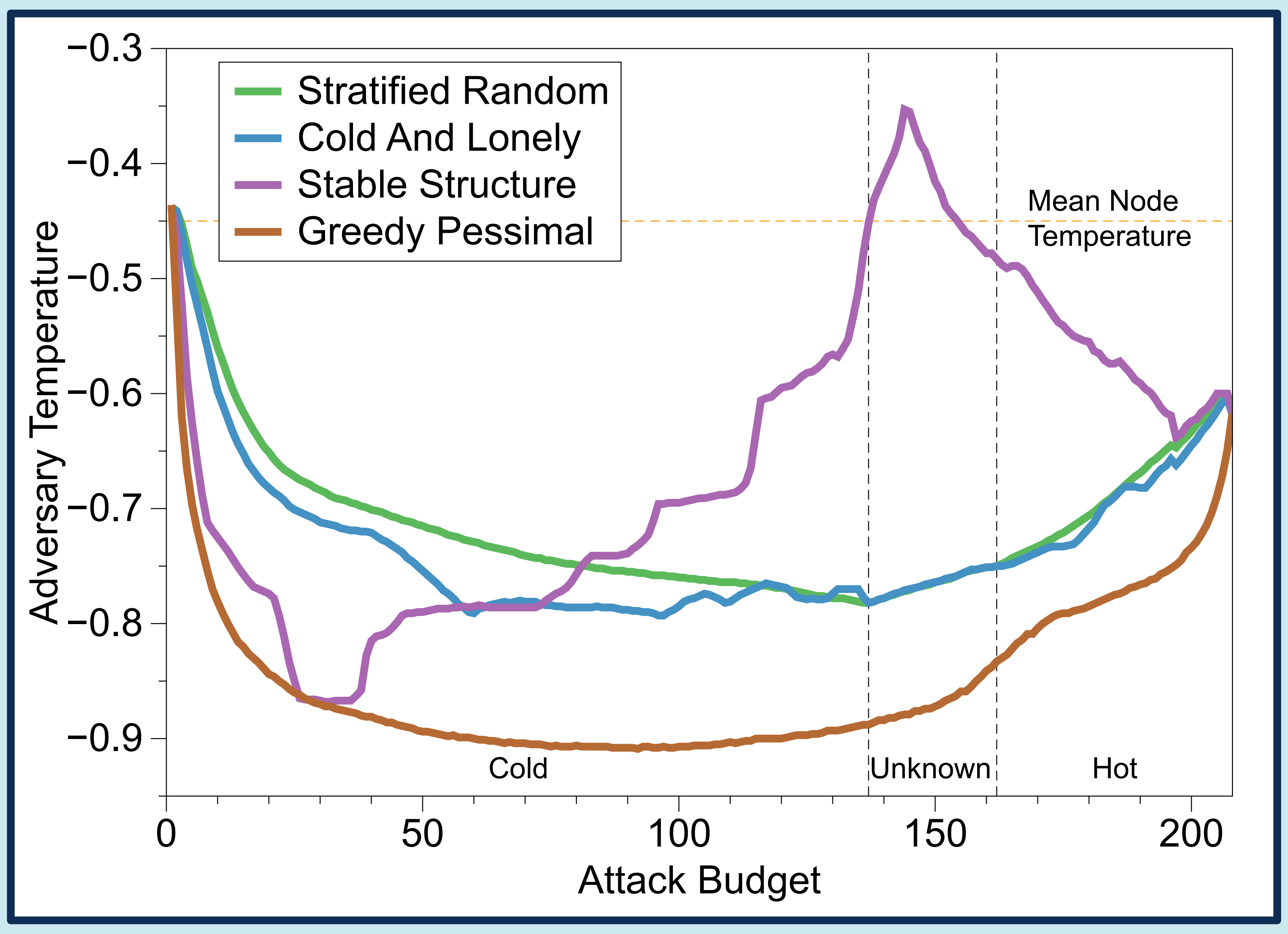


Adversaries may know which algorithms we use for critical applications. By altering public input data, adversaries could sabotage our analyses.



Given a graph with some bad/hot and good/cold labeled nodes, an adversary seeks to move to a colder community than his natural community.

Attacks



In an attack, the adversary defines an order to add edges to all other nodes in the graph. A better attack moves the adversary node to colder communities faster.

A **random attack** (rr; essentially flat at mean temp; not shown) creates a random order of the other nodes in the graph. A **stratified random attack** (sr) randomly orders cold nodes first, then unlabeled nodes, finally hot nodes. A **cold and lonely attack** (cal) sorts nodes by temperature first and increasing degree within temperature bands (random when temperature and degree same). A **stable structure attack** (ss) identifies sets of nodes that are together across Louvain random seeds; orders these stable structures by increasing temperature (random within community); finally, orders remaining nodes as stratified random. A **greedy pessimal attack** (gp) exhaustively identifies which node to next attach to by finding which most decreases the adversary's temperature.

Defenses

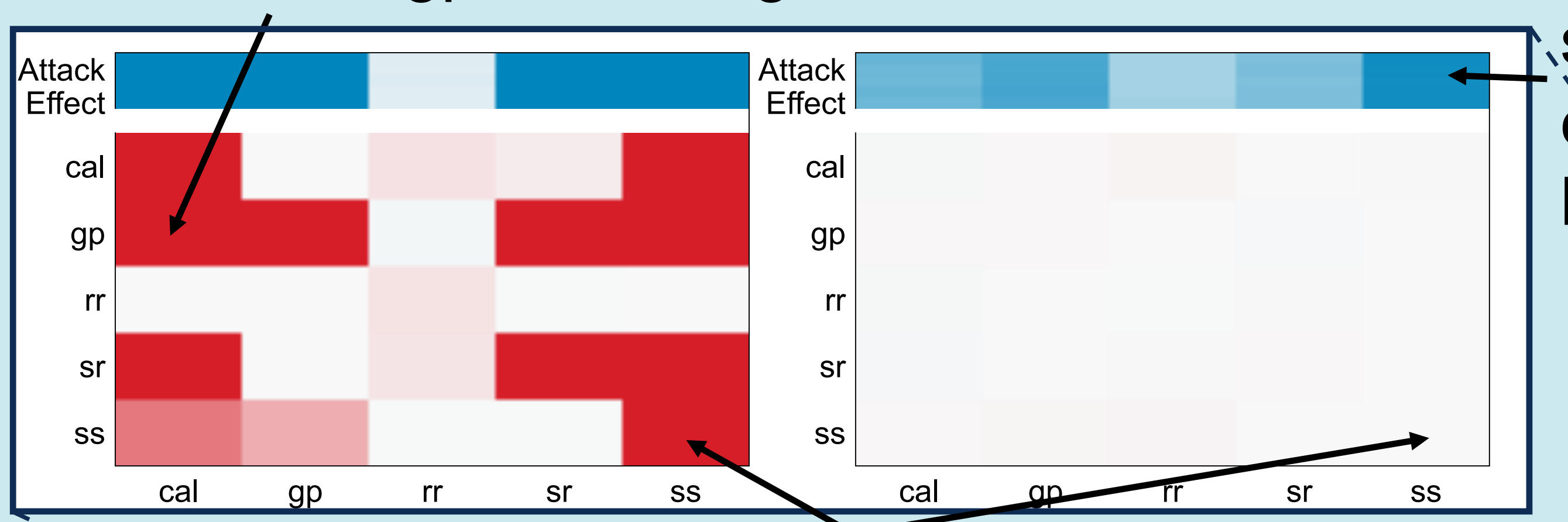
We use machine learning to identify attack edges. To do so, we developed a technique for generating statistically similar graphs to a single graph sample. It approximately maintains nodes' community assignments, degrees, connections within/between communities, and connections with nodes by temperature attribute. Using this model, we generated 100 similar training graphs to our one test graph.

We attacked our training graphs with some adversary node and attack; adding 20 attack edges. We extracted features for all edges in the attacked graph, and trained an ensemble of decision trees to identify the inserted edges. We testing against our original graph as corrupted by some adversary node and attack. We merged all 100 training graphs' features for adversary-node/attack pair into a single model.

Edge features for the machine learning defense model:

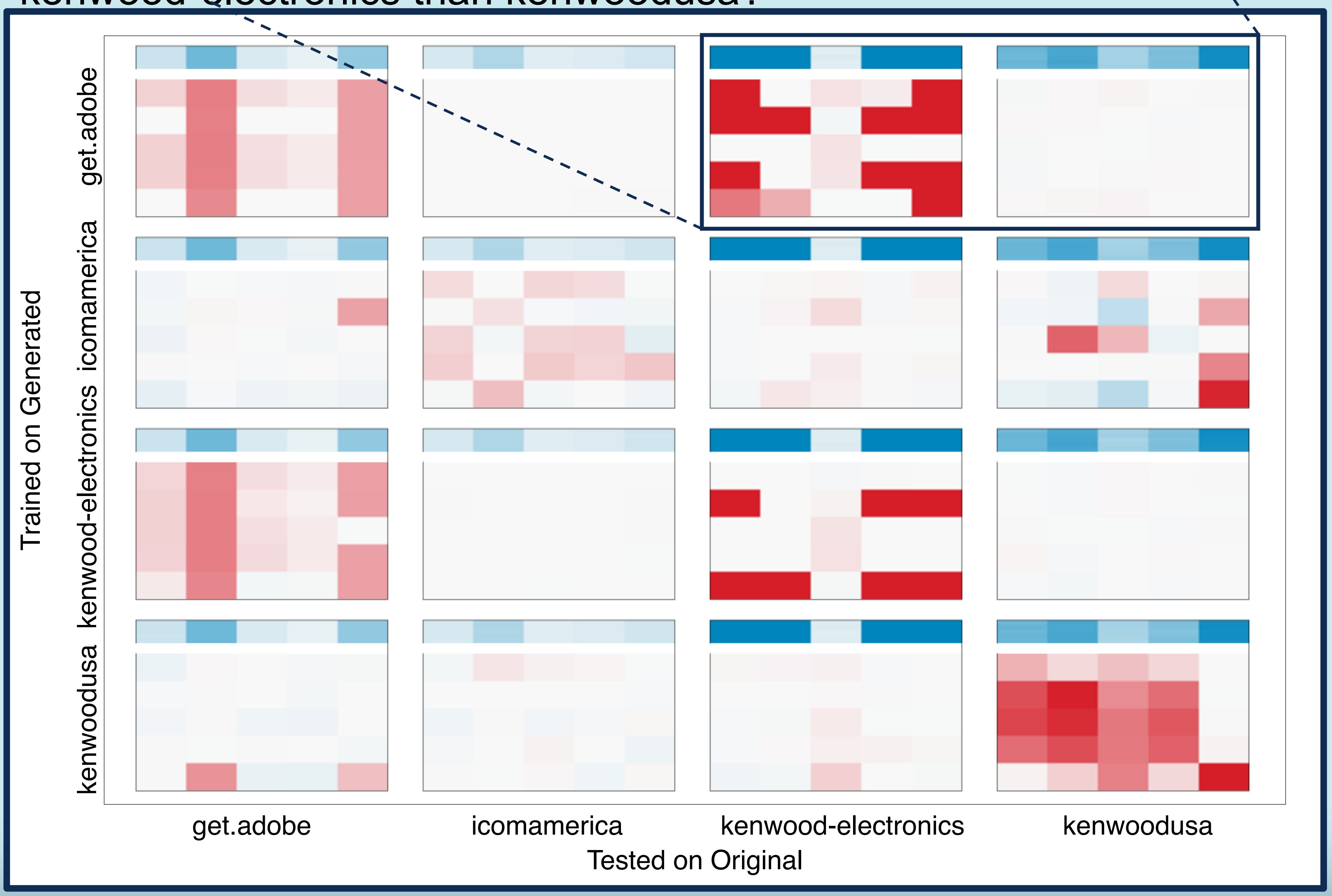
- Number and density of triangles using edge
- Edge endpoints' neighbors - Jaccard similarity
- Source/dest/edge betweenness centrality
- Percentage of Louvain runs that edge is within community
- Source/dest label (each; and do-match?)
- Source/dest eccentricity with/without edge
- Source-to-dest distance without edge
- Source/dest/(src+dest) community temperature (mean, stdev, max)
- abs(source-dest) community temperature (mean, stdev, max)

Trained on gp, tested against cal



ss attack most effective by kenwoodusa node

Continuing research: Why was defense against ss so much better for kenwood-electronics than kenwoodusa?


Trained on Generated
get.adobe
icomamerica
kenwood-electronics
kenwoodusa
get.adobe
icomamerica
kenwood-electronics
kenwoodusa
Tested on Original