Mitigating Strategic Attack Diffusion in Security Games
Supplementary Materials

Paper #2686

Proof of Theorem 1

The decision version of the optimization problem is the following: given a network $G = (V, E)$, a distribution of security resources ϕ, a seed node v_S, a time limit T and a value $r^* \in \mathbb{N}$ expressing the number of nodes to be activated within time limit, does there exist a sequence of nodes $\gamma^* \in \Gamma(V)$ such that $\eta(\gamma^*, T) \geq r^*$.

This problem clearly is in NP, as given a solution, *i.e.*, a sequence $\gamma^* \in \Gamma(V)$, we can compute the time of activation of every node in the sequence and verify whether $\eta(\gamma^*, T) \geq r^*$ in polynomial time.

The main idea of the NP-hardness proof is as follows. We will show a reduction from the NP-complete 3-Set Cover problem. We build a network that reflects the structure of a given 3-Set Cover problem instance and use it as an input for the Optimal Attack problem. Finally, we show that an optimal solution of the Optimal Attack problem corresponds to a solution of the given instance of the 3-Set Cover problem.

An instance of the NP-complete 3-Set Cover problem is defined by a universe $U = \{u_1, \ldots, u_m\}$, a collection of sets $S = \{S_1, \ldots, S_k\}$ such that $\forall_j S_j \subset U$ and $\forall_j |S_j| = 3$, and an integer $b \leq k$. The goal is to determine whether there exist b elements of S the union of which equals U. In what follows let $\sigma(u_i) = |\{S_j \in S : u_i \in S_j\}|$, *i.e.*, $\sigma(u_i)$ is the number of sets in S that contain u_i.

First, let us create a network G, reflecting the structure of the given instance of the 3-Set Cover problem, as shown in Figure 1:

- **The set of nodes:** For every $S_i \in S$, we create a single node, denoted by S_i. For every $u_i \in U$, we create a single node, denoted by u_i, as well as $4k$ nodes $a_{i,1}, \ldots, a_{i,k}$. Moreover, for every $u_j \in U$ and every $S_i \in S$ such that $u_j \in S_i$ we create a single node $l_{i,j}$. Additionally, we create a single node v_S.
- **The set of edges:** For every node $S_i \in S$ we create an edge (S_i, v_S). For every node $a_{i,j}$ we create an edge $(a_{i,j}, u_i)$. Finally, for every node $l_{i,j}$ we create two edges, $(S_i, l_{i,j})$ and $(u_j, l_{i,j})$.

Let ϕ^* be a distribution of security resources such that:

- $\phi^*(v_S) = 0$,
- $\phi^*(a_{i,j}) = 0$ for every node $a_{i,j}$,
- $\phi^*(u_i) = k - \sigma(u_i)$ for every node u_i,

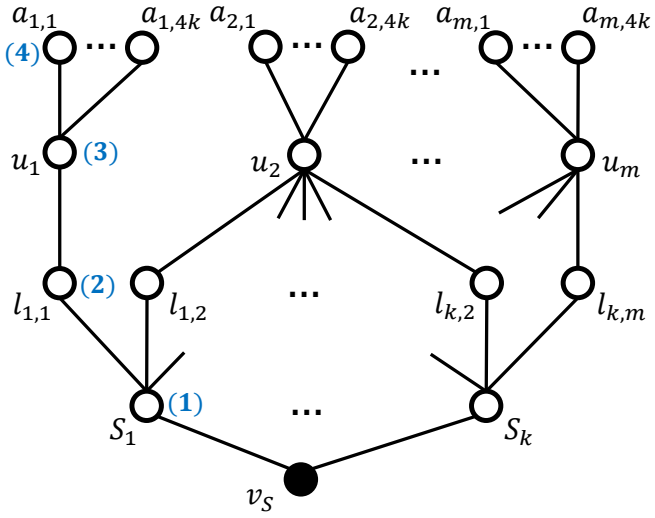

Figure 1: The network used to reduce the 3-Set Cover problem to the Optimal Attack problem. The seed node v_S is marked black. Blue numbers in parenthesis show order of nodes that have to be activated before activating node $a_{1,1}$.

- $\phi^*(S_i) = 20mk$ for every node S_i,
- $\phi^*(l_{i,j}) = 20mk + 2$ for every node $l_{i,j}$.

This particular distribution of security resources gives each u_i node the same probability of activation (assuming an equal number of activated neighbours), and gives each of the nodes S_i and $l_{i,j}$ the maximal probability of activation that is lower than the minimal probability of activation of nodes u_i, *i.e.*, activating node S_i or $l_{i,j}$ always takes more time than activating node u_i, no matter what the number of active neighbours is. This will turn out to be important later in the proof.

Moreover, let $T^* = (b+m)(20mk+4)+9mk+1$. Finally, let $r^* = b + (4k + 2)m$. These are the time and number of activated nodes of a strategy corresponding to a solution to the 3-Set Cover problem.

Now, consider the instance of problem of Optimal Attack in the form of $(G, \phi^*, v_S, T^*, r^*)$. We will now show that an optimal solution to this instance corresponds to an optimal solution to the 3-Set Cover problem.

First, notice that the following holds for the times of activation of nodes in V:

- $\tau(a_{i,j}) = 1$ for every node $a_{i,j}$, as we have $d(a_{i,j}) + \phi^*(a_{i,j}) = 1$ and $|N(a_{i,j}) \cap I| \leq 1$;
- $\tau(u_i) \leq 5k$ for every $u_i \in U$, as we have $d(u_i) + \phi^*(u_i) = 5k$ and $|N(u_i) \cap I| \geq 1$;
- $5mk + 1 \leq \tau(S_i) \leq 20mk + 4$ for every $S_i \in S$, as we have $d(S_i) + \phi^*(S_i) = 20mk + 4$ and $1 \leq |N(S_i) \cap I| \leq 4$;
- $10mk + 2 \leq \tau(l_{i,j}) \leq 20mk + 4$ for every node $l_{i,j}$, as we have $d(l_{i,j}) + \phi^*(l_{i,j}) = 20mk + 4$ and $1 \leq |N(l_{i,j}) \cap I| \leq 2$.

We now show that if there exists a solution $S^* \subset S$ to the given instance of the 3-Set Cover problem, then there exists a solution γ^* to the constructed instance of the Optimal Attack problem such that $\eta(\gamma^*, T) = r^* = b + (4k + 2)m$. Indeed, we can construct such solution by activating every node $S_i \in S^*$ (b nodes activated in time $b(20mk + 4)$), choosing for every $u_j \in U$ node $S_i \in S^*$ such that $u_j \in S_i$ and activating nodes $l_{i,j}$ and u_j ($2m$ nodes activated in time $(20mk+4)m+5mk$), and finally activating all nodes $a_{i,j}$ ($4mk$ nodes in time $4mk$).

In what follows, let γ^*_T denote the maximum prefix of γ^* such that $\tau(\gamma^*_T) < T$. Now, we have to show that if there exists a solution γ^* to the constructed instance of the Optimal Attack problem, then there also exists a solution $S^* \subset S$ to the given instance of the 3-Set Cover problem. To this end, we will now show that for the prefix γ^*_T of any such solution γ^* it must be that for every $u_j \in U$ there exists a node $S_i \in \gamma^*_T$ such that $u_j \in S_i$ and $|\gamma^*_T \cap S| \leq b$. Notice that when this is the case we can obtain the solution to the given instance of the 3-Set Cover problem by taking $S^* = \gamma^*_T \cap S$.

First, notice that any sequence γ^*_T that does not contain all nodes u_i cannot have the required number of r^* activated nodes within time limit. Assume to the contrary, that there exists such a sequence. Since there exists node u_i that is not activated, neither of the nodes $a_{i,j}$ are activated. Therefore this sequence has to activate $4k + 1$ of the nodes S_i or $l_{i,j}$ instead. However, there are only $4k$ nodes S_i and $l_{i,j}$. Hence, sequence γ^*_T that is a solution of the constructed instance of the Optimal Attack problem must activate all nodes u_j. In order to activate given node u_j we need to activate at least one node $l_{i,j}$, and in order to achieve that we need to activate node S_i. Therefore, because of the way we constructed the network, for every $u_j \in U$ there exists a node $S_i \in \gamma^*_T$ such that $u_j \in S_i$.

Now we need to show that for γ^*_T that is a solution of the constructed instance of the Optimal Attack problem we have $|\gamma^*_T \cap S| \leq b$, *i.e.*, that any sequence being solution cannot activate more than b nodes S_i within time limit. As shown above, γ^*_T has to activate all u_j nodes. Notice that in order to activate u_j we have to activate a node $l_{i,j}$ with only one activated neighbour (its only other neighbour being u_j). Activating m such nodes takes $m(20mk + 4)$ time. Hence, in order to activate more than b nodes S_i within time limit we would have to do it in time shorter than $T^* - m(20mk+4) = b(20mk + 4) + 9mk$ (as we still have to activate nodes u_j). Now, activating node S_i (including time necessary to activate neighbouring nodes $l_{i,j}$) takes time:

- $20mk + 4$ when activating with only one active neighbour v_S;
- $\frac{20mk+4}{2} + (20mk + 4)$ when activating with two active neighbours;
- $\frac{20mk+4}{3} + 2(20mk+4)$ when activating with three active neighbours;
- $\frac{20mk+4}{4} + 3(20mk+4)$ when activating with four active neighbours.

This is because node $l_{i,j}$ in order to be used to speed up the activation of node S_i has to be activated beforehand, in time $20mk + 4$. Hence, activating a single node S_i takes at least $20mk + 4$ time and it is not possible to activate more than b nodes S_i in time $b(20mk + 4) + 9mk$.

This implies that the optimal solution to the constructed instance of the Optimal Attack problem must correspond to the optimal solution to the given instance of the 3-Set Cover problem, thus concluding the proof.

Proof of Theorem 2

Since the network is a clique, the activation time of i-th node in a sequence γ is $\frac{n-1+\phi(\gamma_i)}{\max(1,i-1)}$ (as γ_i has $n-1$ neighbors, $i-1$ of which has been already activated).

First we prove our claim about the strategy of the attacker.

We prove it by contradiction. Assume to the contrary, that in an optimal attack sequence γ we have $\phi(\gamma_i) > \phi(\gamma_j)$ for $i < j$ such that $i > 1$ and $j > 2$ (notice that we can always swap first and second element of the sequence as the sum of their activation times remains the same). However, such sequence can be improved by swapping elements i and j. Let τ_{ij} be time of activation of γ_i and γ_j before the swap, and τ_{ji} after the swap (activation times of all other nodes remain the same after the swap). We have (notice that $\max(1, i-1) = i-1$ and $\max(1, j-1) = j-1$ for $i > 1$ and $j > 2$):

$$\begin{aligned}\tau_{ji} - \tau_{ij} =& \left(\frac{n-1+\phi(\gamma_j)}{i-1} + \frac{n-1+\phi(\gamma_i)}{j-1}\right) \\ &- \left(\frac{n-1+\phi(\gamma_i)}{i-1} + \frac{n-1+\phi(\gamma_j)}{j-1}\right) \\ =& \frac{\phi(\gamma_i) - \phi(\gamma_j)}{j-1} - \frac{\phi(\gamma_i) - \phi(\gamma_j)}{i-1} < 0.\end{aligned}$$

Therefore, in an optimal sequence for $i < j$ such that $i > 1$ and $j > 2$ we always have $\phi(\gamma_i) \leq \phi(\gamma_j)$.

Now we move to proving our claim about the strategy of the defender.

We prove it by contradiction. Assume to the contrary, that in an optimal distribution of security resources ϕ we have $i > 1$ such that $\phi(\gamma^*_i) \neq \phi(\gamma^*_{i+1})$ where γ^* is an optimal attack sequence (again, notice that we can always swap first and second element of the attack sequence as the sum of their activation times remains the same). Since γ^* is an optimal attack sequence we have that $\phi(\gamma^*_i) < \phi(\gamma^*_{i+1})$. However, such ϕ can be improved by setting $\phi(\gamma^*_i)$ and $\phi(\gamma^*_{i+1})$ to their average. Let τ_1 be time of activation of γ^*_i and γ^*_{i+1} before the change, and τ_2 after the change (activation times of all other

nodes remain the same after the change). We have:

$$\begin{aligned}\tau_2 - \tau_1 =& \left(\frac{n-1+\frac{\phi(\gamma_i^*)+\phi(\gamma_{i+1}^*)}{2}}{i-1} + \frac{n-1+\frac{\phi(\gamma_i^*)+\phi(\gamma_{i+1}^*)}{2}}{i}\right) \\ &- \left(\frac{n-1+\phi(\gamma_i^*)}{i-1} + \frac{n-1+\phi(\gamma_{i+1}^*)}{i}\right) \\ =& \frac{\phi(\gamma_{i+1}^*)-\phi(\gamma_i^*)}{2(i-1)} - \frac{\phi(\gamma_{i+1}^*)-\phi(\gamma_i^*)}{2i} \\ =& \frac{\phi(\gamma_{i+1}^*)-\phi(\gamma_i^*)}{2i(i+1)} > 0.\end{aligned}$$

Therefore, in an optimal sequence distribution of security resources we have $\phi(\gamma_i^*) = \phi(\gamma_{i+1}^*)$ for $i > 1$.

Proof of Theorem 3

First we prove our claim about the strategy of the attacker.

Notice that node a has to be either first or second node in the attack sequence, as the attacker either starts with it, or it is the only neighbour of the starting node (if attacker starts with one of the peripheral nodes). Hence, it has activation time of $\tau(a) = n - 1 + \phi(a)$.

Similarly, any node b_i is either first in the sequence or is activated when exactly one of its neighbours (central node a) is active. Hence, its activation time is $\tau(b_i) = 1 + \phi(b_i)$.

If it is not possible to activate more than one node within time limit, the sequence γ^* is optimal because the first node has the lowest time of activation in the entire network.

Now consider a case where it is possible to activate at least two nodes within time limit. Assume to the contrary, that in an optimal attack sequence γ we have b_j activated before b_i for $i < j$ (as mentioned above, node a has to be either on first or second position in the sequence and we can swap these nodes without changing the total time of activation). However, such sequence can be improved by swapping elements b_i and b_j. The argument follows the same as in the proof of Theorem 2.

Now we move to proving our claim about the strategy of the defender.

First, consider a case in which it is possible not to let attacker activate even a single node, *i.e.*, $\forall_{v \in V} \tau(v) \geq T$. It is then optimal for the defender to have the same activation time for all nodes (as the attacker will pick the one with lowest activation time as the first in sequence). Hence, we have $\tau(a) = \phi^*(a) + n - 1 = \phi^*(b_i) + 1 = \tau(b_i)$ and $\phi^*(a) + (n-1)\phi^*(b_i) = \Phi$. After solving this set of equations we get $\phi^*(a) = \frac{\Phi-(n-1)(n-2)}{n}$ and $\phi^*(b_i) = \frac{\Phi+n-2}{n}$. We then have $\tau(a) = \tau(b_i) = \frac{\Phi+2(n-1)}{n}$. Since we consider a case in which it is possible not to let attacker activate even a single node, we need to have $\tau(a) = \tau(b_i) = \frac{\Phi+2(n-1)}{n} \geq T$.

Notice that it is possible that the defender does not have enough defense resources to make time of activation of all nodes equal (when $\Phi < (n-1)(n-2)$). She should then spread them uniformly among peripheral nodes, to maximize the minimal time of activation in the network. We then have $\phi^*(a) = 0$ and $\phi^*(b_i) = \frac{\Phi}{n-1}$. Minimal time of activation in the network is the $\tau(b_i) = \frac{\Phi+n-1}{n-1}$ and in order for this strategy to be optimal we need to have $\tau(b_i) = \frac{\Phi+n-1}{n-1} \geq T$.

Now, consider a case in which the attacker is able to activate $k \geq 1$ nodes. One of the first two nodes in the sequence has to be a. Hence, assigning all security resources to a either maximizes the time required to activate available number of nodes if $k \geq 2$ (as their total activation time is $\phi(a) + \sum_{i=1}^{k-1} \phi(b_i)$), as well as it maximizes the time that would be necessary to activate second node in the sequence if $k = 1$.

Proof of Theorem 4

We present the pseudocode of algorithm computing optimal attacker's strategy on a tree starting from a given node as Algorithm 1. It performs exhaustive search of all possible strategies of the attacker by traversing the tree in a bottom-up fashion. In the pseudocode V_{botup} denotes sequence of nodes in V in a bottom-up order (when v_S is a root), $c(v)$ denotes the sequence of children of v, $c_i(v)$ denotes the i-th node in this sequence, and $t(v)$ denotes the size of subtree with v as the root.

One can notice that in case of a tree every node other than the source node have exactly one active neighbor at the moment of activation. Hence, its time of activation is always $d(v) + \phi(v)$. In what follows, we will call this value the *weight* of node v. The task of finding the optimal strategy of the attacker comes down to finding largest subtree with the sum of weights lower than T. Algorithm 1 achieves this goal by filling tables τ and γ.

Entry $\tau[v, k]$ contains weight of the lightest subtree with k nodes rooted at v, while $\gamma[v, k]$ contains nodes in this lightest subtree (in the order of activation). Computing values of $\tau[v, 0]$ and $\tau[v, 1]$ (as well as $\gamma[v, 0]$ and $\gamma[v, 1]$) is trivial. In case node v is not a leaf, we compute $\tau[v, k]$ and $\gamma[v, k]$ for $k > 1$ by filling tables y and q.

Entry $y[m, k]$ contains weight of the lightest subtree with k nodes rooted at v, constructed only using descendants of first m children of v. Entry $q[m, k]$ contains nodes in this lightest subtree (in the order of activation). In lines 18-23 we check all possible values of $y[m, k]$ by taking i nodes from the first $m - 1$ children of v (and v itself) and adding to them $k - i$ nodes from the subtree of the m-th child of v.

Instructions in lines 11-12 as well as instructions in lines 25-26 are run $\mathcal{O}(n^2)$ times, since loop in line 4 is run exactly n times and for any v we have $t(v) \leq n$. One can notice that the body of loop in line 14 is run exactly $n - 1$ times in total (because it is a loop over children of node v and every node other than v_S has exactly one parent). Since both $l \leq n$ and $k \leq n$, instructions in lines 19-23 are run $\mathcal{O}(n^3)$ times. Hence, the time complexity of Algorithm 1 is $\mathcal{O}(n^3)$. Memory complexity of Algorithm 1 is $\mathcal{O}(n^2)$ if we represent sequences in tables γ and q as trees visited in prefix order.

Experimental Results for Large Networks

Figure 2 presents our results for networks with 1000 nodes, taken as an average over 100 simulations. Colored areas represent 95% confidence intervals. Because of the time and

memory complexities we are unable to compute the optimal solution for networks of this size. However, we present results of the heuristic algorithms to observe how trends in them change with the size of the network.

Most of our observations made for the smaller networks remain valid. One notable difference is a great increase of the effectiveness of High Degree strategy. In most cases it now outperforms Uniform strategy and the difference is especially striking in case of the preferential attachment networks. In this type of networks the expected degree of hubs greatly increases with size of the network, hence attacker's advantage from activating a hub and gaining access to large part of the network is also much higher. Because of this, focusing defender's efforts on protecting high degree nodes significantly improves effectiveness of the defense.

Another difference is that the Epsilon-Decreasing strategy, usually achieving poor performance for small networks, in large networks is doing better in comparison to Mixed and Epsilon-First strategies. In many cases it has better results than pure Conformist or pure Greedy strategies, depending on the defender's strategy.

Experimental Results with Mixed-Integer Linear Programming

To test the effectiveness of mixed-integer linear programming in finding the optimal strategy of the defender we perform experiments on randomly generated networks. In order to avoid selecting arbitrary set of strategies $\hat{\Gamma}$ available to the attacker, we allow the attack to choose any strategy. However, with these assumptions, we were able to solve mixed-integer linear programming instance efficiently only for very small networks. A potential idea for future work is studying whether there exists a way of finding a smaller set $\hat{\Gamma}$ that is still guaranteed to always contain the optimal attacker strategy.

Figure 3 presents our results for networks with 6 nodes, taken as an average over 100 simulations. Colored areas represent 95% confidence intervals. Mixed-integer linear programming were solved using *lp_solve* solver version 5.5.2.5.

As it can be seen, distribution of security resources computed using mixed-integer linear programming is considerably more successful in mitigating the attack than any other defense strategy. This suggests that there is still room for improvement in terms of heuristic defender strategies to us in networks where the solution to the mixed-integer linear programming formulation cannot be computed by solvers.

Another difference in comparison to results for larger networks is that all considered strategies of the attacker achieve very similar results.

Algorithm 1: Finding the optimal attack sequence for a tree.

Input: Tree $G = (V, E)$, distribution of security resources ϕ, time limit T, starting node v_S.
Output: Optimal attack sequence starting with node v_S

for $v \in V$ **do**
  for $k \in \langle 1, \ldots, n \rangle$ **do**
    $x[v, k] \leftarrow \infty$
for $v \in V_{botup}$ **do** // loop over all nodes in bottom-up order
  $\tau[v, 0] \leftarrow 0$
  $\gamma[v, 0] \leftarrow \emptyset$
  $\tau[v, 1] \leftarrow d(v) + \phi(v)$
  $\gamma[v, 1] \leftarrow \langle v \rangle$
  if $d(v) > 1$ **then** // if v is not a leaf aggregate results from children
    for $k \in \langle 1, \ldots, t(c_1(v)) \rangle$ **do** // choosing targets only from the subtree of the first child
      $y[1, k] \leftarrow d(v) + \phi(v) + \tau[c_1(v), k - 1]$
      $q[1, k] \leftarrow \langle v \rangle \cup \gamma[c_1(v), k - 1]$
    $l \leftarrow t(c_1(v)) + 1$
    for $m \in \langle 2, \ldots, |c(v)| \rangle$ **do** // choosing nodes from the subtrees of the first m children
      $l \leftarrow l + t(c_m(v))$
      for $k \in \langle 1, \ldots, l \rangle$ **do** // loop over the number of attacked nodes
        $y[m, k] \leftarrow \infty$
        for $i \in \langle 1, \ldots, k \rangle$ **do** // loop over the number of attacked nodes we select from the subtrees of the first $m - 1$ children
          $\tilde{y} \leftarrow y[m - 1, i] + \tau[c_m(v), k - i]$
          if $\tilde{y} < y[m, k]$ **then**
            $y[m, k] \leftarrow \tilde{y}$
            $q[m, k] \leftarrow$
              $q[m - 1, i] \cup \gamma[c_m(v), k - i]$
    for $k \in \langle 2, \ldots, t(v) \rangle$ **do** // transferring results to tables τ and γ
      $\tau[v, k] \leftarrow y[|c(v)|, k]$
      $\gamma[v, k] \leftarrow q[|c(v)|, k]$
return $\max_{k:\tau[v_S,k]<T} \gamma[v_S, k]$

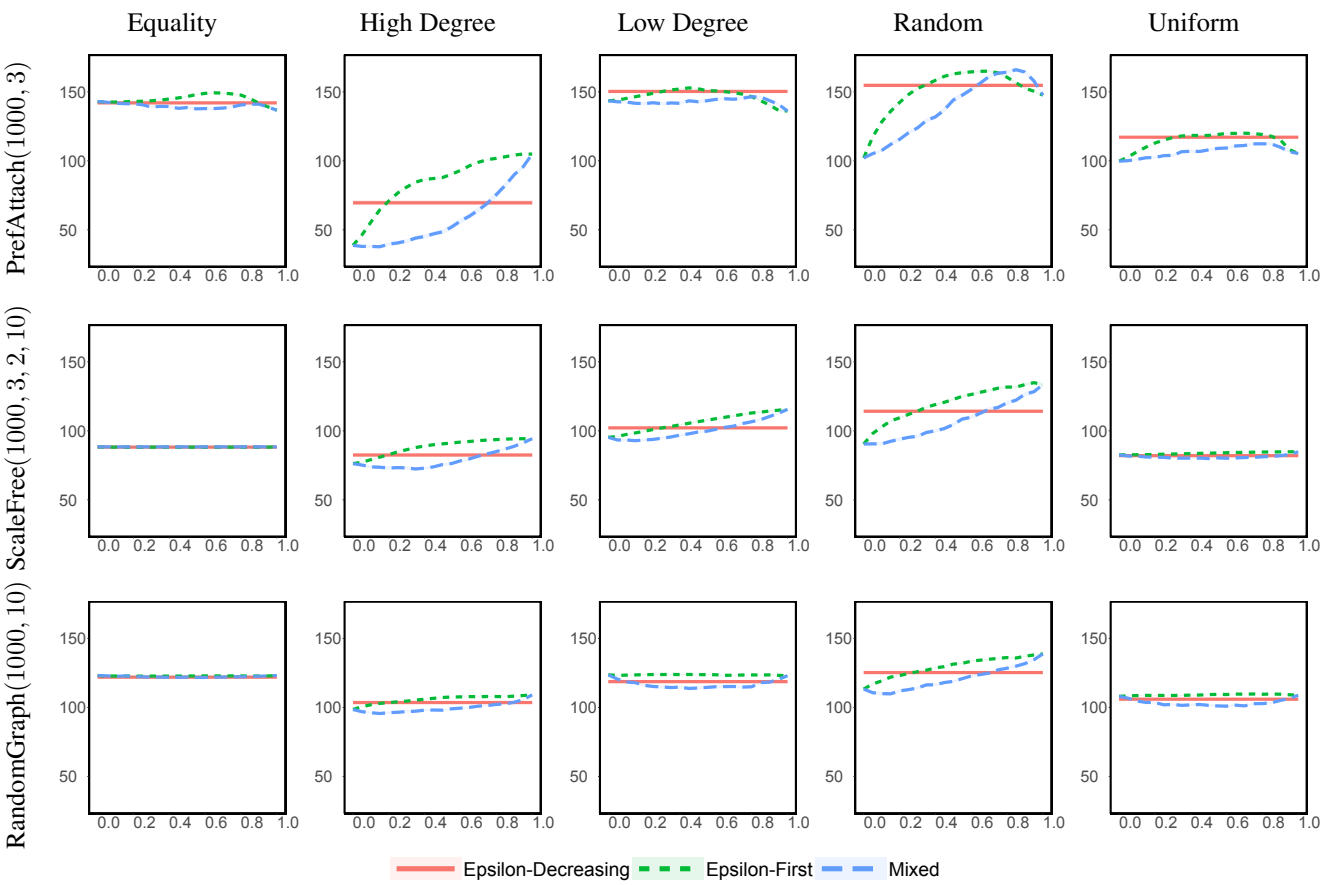

Figure 2: Comparison of the results of using various strategies of the attacker for large networks with 1000 nodes. Each row represents model of network generation, each column represents different strategy of the defender. Each line in a plot represents results of a different strategy of the attacker. The y-axis represents the number of activated nodes after the process finished, which is used as a performance measure, the lower the better for the defender. In case of the Mixed strategy the x-axis represents the probability of using Greedy (while the alternative is using Conformist). In case of the Epsilon-First strategy the x-axis represents the value of $1 - \epsilon$. Notice that this implicates that for $x = 0$ Mixed and Epsilon-First are equal to pure Conformist, while for $x = 1$ they are equal to pure Greedy. Also, notice that for the Epsilon-Decreasing strategy all lines are flat, as these strategies are not parameterized, and the line simply indicates performance of the strategy. The scale is fixed to make plots easier to compare.

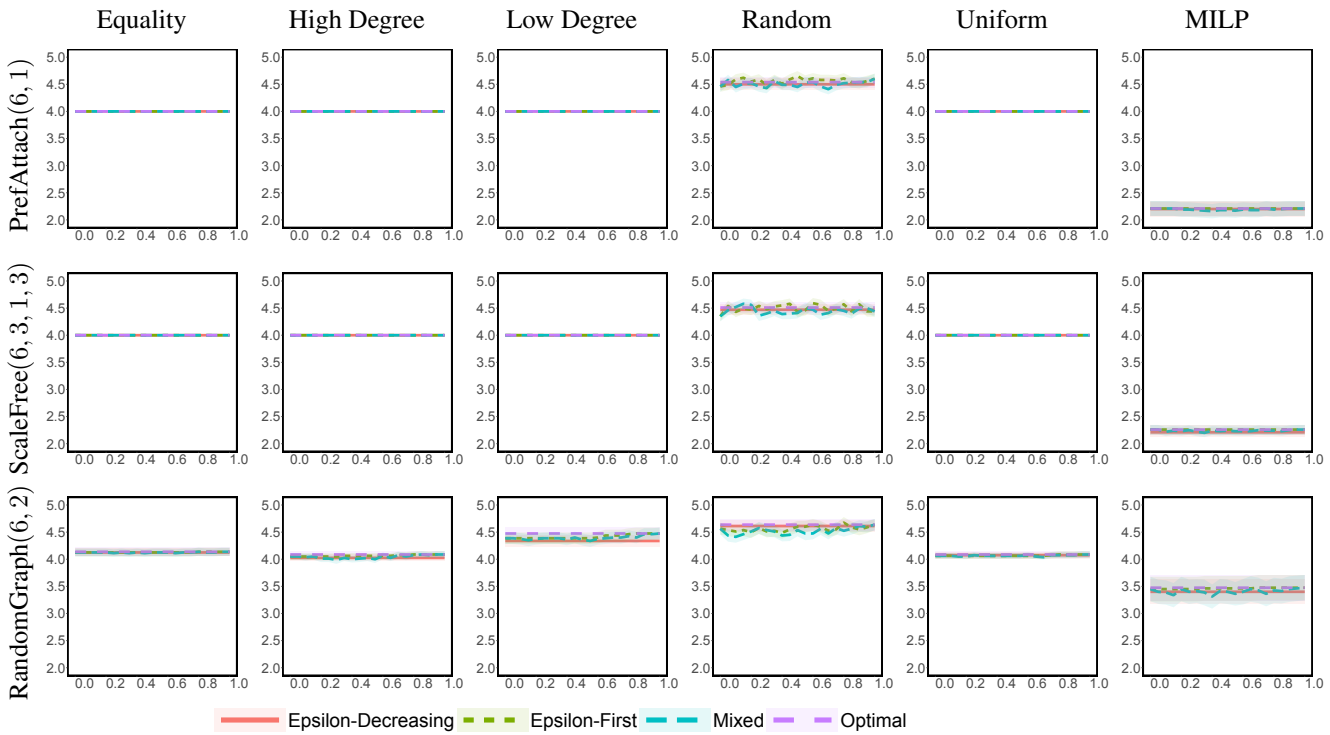

Figure 3: Comparison of the results of using various strategies of the attacker for very small networks with 6 nodes. Each row represents model of network generation, each column represents different strategy of the defender. Each line in a plot represents results of a different strategy of the attacker. The y-axis represents the number of activated nodes after the process finished, which is used as a performance measure, the lower the better for the defender. In case of the Mixed strategy the x-axis represents the probability of using Greedy (while the alternative is using Conformist). In case of the Epsilon-First strategy the x-axis represents the value of $1 - \epsilon$. Notice that this implicates that for $x = 0$ Mixed and Epsilon-First are equal to pure Conformist, while for $x = 1$ they are equal to pure Greedy. Also, notice that for the Epsilon-Decreasing strategy all lines are flat, as these strategies are not parameterized, and the line simply indicates performance of the strategy. The scale is fixed to make plots easier to compare.