Hide Exposures by Removing Mastermind's External Sources on Social Network (Student Abstract)

Nilanjana Saha[1], **Narayan Changder**[2], **Redha Taguelmimt**[3], **Samir Aknine**[4], **Animesh Dutta**[1]
[1]National Institute of Technology Durgapur, West Bengal, India.
[2]Institute for Advancing Intelligence (IAI), TCG CREST, Kolkata, West Bengal, India. [3] Clermont Auvergne University, Clermont Auvergne INP, CNRS, LIMOS, F-63000 Clermont-Ferrand, France.[4] LIRIS, Lyon 1 University, Lyon, France.
nilanjanasaha.2812@gmail.com, narayan.changder@tcgcrest.org, redha.taguelmimt@gmail.com, samir.aknine@univ-lyon1.fr, animesh@cse.nitdgp.ac.in

Abstract

On social media, it is easy to see how people are connected and find the leader, or mastermind of a network. The mastermind is responsible for the planning of the activities in the network. Hiding the mastermind is important to carry out these activities. This raises the question for the mastermind: *How effectively can the mastermind hide his connections to avoid being found?* We propose an efficient heuristic algorithm called HERMES (Hide Exposures by Removing Mastermind's External Sources) to address this. Experiments on Facebook and Google networks show that HERMES hides the mastermind more effectively than the state-of-the-art, achieving time gains of 103 and 1397 seconds, respectively, and improving influence value by up to 11.11%.

The HERMES Algorithm

Let $G = (V, E)$ denote a social network, where V is the set of nodes (users) and E the edges (connections between users). A *core community* $C_i \subseteq V$ is a densely connected subgraph where each node in C_i has a minimum degree of connections to other nodes within the same community. A *mastermind* $M \in V$ is the most influential node within C_i. In an online social network like Facebook, a mastermind spreading misinformation connects with a core group of influential users who amplify the fake news. If the mastermind interacts directly with users outside the core group, investigators can trace these actions back to them. However, by avoiding direct interactions and relying on the core group to spread the misinformation, the mastermind remains hidden, complicating detection. The state-of-the-art algorithm for hiding a mastermind within a social network is ROAM (Waniek et al. 2018). While ROAM focuses on the most influential connections of the mastermind across the entire network, HERMES targets a *core group* within the network. HERMES first identifies a set of communities $\mathcal{S}_c$ in G using the *Louvain community detection algorithm* (cf. Line 1 in Algorithm 1). Next, HERMES ranks each node within each community $C_i \in \mathcal{S}_c$ based on the *degree centrality (DC)*, *closeness centrality (CC)*, and *betweenness centrality (BC)*. The node with the highest centrality value is ranked 1, the second highest is ranked 2, and so on. Ties are resolved randomly. Consider Figure 1(a). HERMES identifies seven communities $C_1, C_2, \ldots, C_7$. Community $C_7 = \{10, 29, 11, 25, 24, 16, 6\}$ is identified as the core community. The centrality values and ranks for each node in C_7 are shown in Table 1. The average rank of a

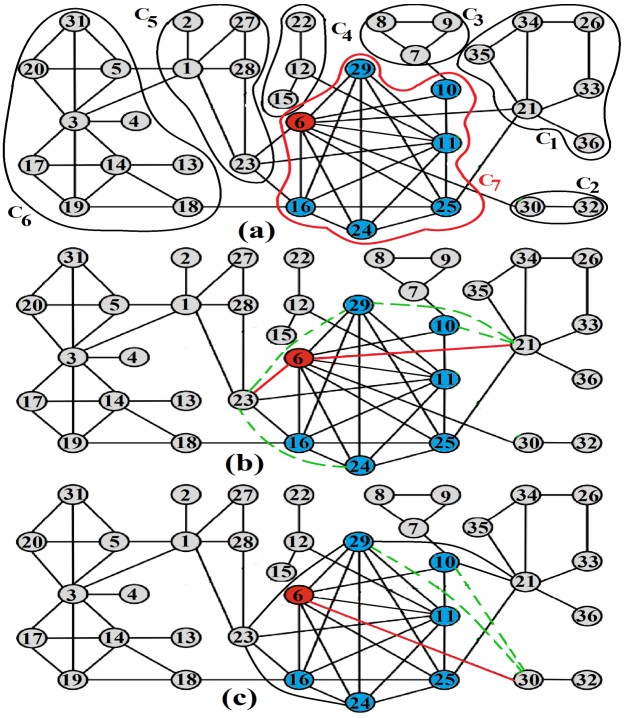

Figure 1: Executing HERMES on the 9/11 terrorist network.

Node	DC(Rank)	CC(Rank)	BC(Rank)
10	0.085 (16)	0.330 (11)	0.161 (6)
29	0.142 (9)	0.336 (7)	0.000 (27)
11	0.228 (2)	0.402 (2)	0.190 (5)
25	0.171 (6)	0.357 (8)	0.059 (14)
24	0.142 (10)	0.336 (4)	0.000 (28)
16	0.199 (3)	0.393 (3)	0.159 (8)
6	0.257 (1)	0.426 (1)	0.389 (1)

Table 1: Centrality value and rank for each node in C_7

Algorithm 1: **HERMES Heuristic**

Input: A graph $G = (V, E)$, the number of iterations $iter$, the number of connection and disconnection nodes b and the number of top d external nodes used in each iteration.
Output: Rank of the mastermind M.

1: Identify the core community C_i in the graph G.
2: Identify $M \in C_i$ based on the centrality measures.
3: Determine the set of external nodes ($NC^{\dagger}$) connected to M but not in C_i
4: **for** $k \leftarrow 1$ to $iter$ **do**
5: Find the top d most influential external nodes $NC_{max} \in NC^{\dagger}$ and remove the links between M and the nodes in NC_{max}. Reconnect these nodes with less influential nodes in C_i.
6: Remove these d external nodes from $NC^{\dagger}$.
7: **end for**

community is determined by summing the ranks its nodes and dividing by the total number of nodes in the community. The community with the lowest average rank across all three centrality measures is the core community. HERMES identifies C_7 as the core community due to its lowest average rank, calculated as follows: For DC, the average rank is $\frac{16+9+2+6+10+3+1}{7} = \frac{47}{7} \approx 6.71$. For CC, it is $\frac{39}{7} \approx 5.57$. For BC, it is $\frac{89}{7} \approx 12.71$. Next, HERMES finds the most influential nodes in C_7 based on the ranks of DC, CC, and BC. The node with the highest rank is identified as the Mastermind M (cf. Line 3 in Algorithm 1). In Figure 1(a), node 6 is identified as the Mastermind, with a rank of 1 for DC, CC, and BC. If no node ranks 1 across all measures, the node with the highest rank in the most measures is chosen. Once the mastermind is identified, HERMES determines the external nodes set $NC^{\dagger}$, which are the mastermind's neighbors outside the core community (cf. Line 3 in Algorithm 1). In Figure 1(a), $NC^{\dagger} = \{21, 23, 30\}$. From $NC^{\dagger}$, the top d most influential external nodes NC_{max} are selected based on DC, CC, and BC values, where $1 \leq d \leq |NC_{max}|$ (cf. Line 5 in Algorithm 1). HERMES operates iteratively (cf. Lines 4-7 in Algorithm 1), selecting the top d external nodes in each iteration. In our example, $d = 2$. HERMES disconnects these nodes from the mastermind and establishes new links with less influential core community members (cf. Line 5 in Algorithm 1). This creates indirect connections, masking the mastermind's involvement. The total number of external nodes involved in both disconnection and connection processes is denoted as b, resulting in $b - 1$ new links being formed. In Figure 1(b), in the first iteration, HERMES identifies 21 and 23 as the top $d = 2$ external nodes. It removes the direct connections between nodes 21 and 23 and the mastermind node 6, creating new indirect links to less influential core members. Edges (6,21) and (6,23) are removed (red edges), and new links (21,29), (21,10) for 21, and (23,24), (23,10) for 23 are formed (green dashed edges). In the next iteration, node 30 is found connected to the mastermind. HERMES removes (6,30) and creates links (30,29) and (30,10), as shown in Figure 1(c). This approach prioritizes the most influential nodes first, with remaining nodes handled in subsequent iterations.

Network	Heuristic	Time	Iter	IC	LT
Facebook (V, E) = (4039, 88234)	ROAM	323 sec	1	0.72	0.09
			2	0.71	0.11
			3	0.70	0.12
	HERMES	220 sec	1	0.71	0.10
			2	0.70	0.11
			3	0.69	0.13
Google (V, E) = (15763, 149456)	ROAM	5784 sec	1	0.52	0.35
			2	0.51	0.37
			3	0.50	0.38
	HERMES	4387 sec	1	0.51	0.36
			2	0.50	0.38
			3	0.48	0.39

Table 2: HERMES vs. ROAM Performance ($b = 3, d = 2$): Time indicates how long algorithm takes to identify M.

Experiments and Discussion: To evaluate HERMES, we analyze centrality rankings with Independent Cascade (IC) and Linear Threshold (LT) influence values (Shakarian et al. 2015). Table 2 shows results for HERMES and ROAM on the Facebook and Google networks. Compared to ROAM, HERMES reduces the time to identify the mastermind by 103 seconds on Facebook and 1397 seconds on Google. This speed improvement is due to HERMES partitioning the network into smaller communities, which reduces the number of nodes involved in centrality calculations and sorting. In contrast, ROAM operates on the entire network, leading to more computationally expensive operations. HERMES achieves an IC value that is 1.38% lower and an LT value that is 11.11% higher than those of ROAM. In iteration 2, the IC value is 1.4% lower, while the LT value is the same. In iteration 3, IC is lower by 1.4%, and the LT value is higher by 8.33%. For Google, in iteration 1, HERMES achieves an IC value that is 1.92% lower and an LT value that is 2.85% higher than ROAM. In iteration 2, the IC value is 1.96% lower, and the LT value is 2.70% higher. By iteration 3, the IC value is 4.00% lower, while the LT value is 2.63% higher. In summary, HERMES consistently achieves lower IC values and higher LT values compared to ROAM, except in iteration 2 for Facebook. The reduction in IC values and the increase in LT values achieved by HERMES make it more difficult for investigators to trace the mastermind.

Acknowledgments

This research work is supported by the Visvesvaraya Ph.D. Scheme Phase-II, MeitY, Govt. of India, under Project ID: PhD-02/2022/35.

References

Shakarian, P.; Bhatnagar, A.; Aleali, A.; Shaabani, E.; and Guo, R. 2015. The independent cascade and linear threshold models. *Diffusion in social networks*, 35–48.

Waniek, M.; Michalak, T. P.; Wooldridge, M. J.; and Rahwan, T. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2): 139–147.