# Information Exposure Maximization Problem with Heuristic and Evolutionary Algorithms

Layheng Hok
*Department of Computer Science and Engineering*
*Southern University of Science and Technology*
Shenzhen, China
12210736@mail.sustech.edu.cn

***Abstract*—This paper explores the Information Exposure Maximization Problem (IEMP) as it applies to reducing echo chambers and filter bubbles in social networks. By representing the network as a graph, we examine the effectiveness of heuristic and evolutionary algorithms in balancing exposure to diverse information. The heuristic approach employs a Monte Carlo-based greedy selection strategy, while the evolutionary algorithm uses a genetic algorithm to optimize influence spread. Results indicate that the heuristic method consistently outperforms the evolutionary algorithm in efficiency and evaluation value, especially for larger datasets. However, the evolutionary algorithm with a halt condition shows promise in reducing computational time. This study underscores the utility of heuristic methods in influence maximization and suggests potential improvements in evolutionary approaches for similar applications.**

***Keywords—evolutionary algorithm, genetic algorithm, heuristic algorithm, optimization***

## I. Introduction

The rise of social networks has led to increased occurrences of echo chambers and filter bubbles, where users predominantly interact with like-minded individuals and viewpoints. This phenomenon restricts exposure to diverse information and contributes to polarization. Addressing this, the Information Exposure Maximization Problem (IEMP) aims to enhance exposure to multiple viewpoints in a network by selecting sets of users that maximize the reach of competing campaigns, thus diversifying the information spread within the network.

This project examines IEMP within a graph-based model of social networks, where users and their connections are represented as nodes and edges. By identifying optimal user sets for two distinct campaigns, we aim to achieve a balanced exposure across the network, potentially countering echo chambers. This report investigates two primary algorithmic approaches: a Monte Carlo-based heuristic strategy and a genetic algorithm-based evolutionary strategy. The purpose of this study is to evaluate the effectiveness and efficiency of these approaches and to assess their suitability for real-world applications in influence maximization and information diversification.

## II. Preliminary

The IEMP is framed as an optimization algorithmic problem aimed at reducing the effects of echo chambers and filter bubbles, where users are frequently limited to similar viewpoints, isolating them within their own information spheres. This study examines a social network represented as a graph $G = (V, E)$ where $V$ represents users (nodes) and $E$ represents social connections (edges) between them. The objective in the IEMP is to identify two sets of users that achieve the highest balanced exposure to diverse information.

The following sections provide essential definitions of terminology and notations used, and a formal definition of the problem.

### A. *Terminology and Notations*

- **Social Network** - $G = (V, E)$ represents the network graph, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of nodes, and $E \subseteq V \times V$ represents the edges between nodes.
- **Campaigns** – $C = \{c_1, c_2\}$ represents two campaigns; each campaign holds its own viewpoint.
- **Initial Seed Set** – $I_i \subseteq V$, $i \in \{1, 2\}$ represents the initial seed set for campaign $c_i$ .
- **Balanced Seed Set** – $S_i \subseteq V$, $i \in \{1, 2\}$ represents the target seed set needed to find for each campaign $c_i$.
- **Budget** – $k$ represents the size's upper bound of the two balanced seed sets; that is, $|S_1| + |S_2| \leq k$.
- **Diffusion Probability** – $P_i = \{p^i_{(u,v)} \mid (u, v) \in E\}$, $i \in \{1, 2\}$ represents the edge weight associated with campaign $c_i$ where $p^i_{(u,v)}$ represents the probability of node $u$ activating node $v$ under each campaign $c_i$.
- **Diffusion Model** – $M$ represents the stochastic process for the seed $U_i = I_i \cup S_i$, which initiates the spread of information on graph $G$. It is assumed that information from the two campaigns spreads through the network following the independent cascade (IC) model, with each campaign's message spreading independently- this is known as heterogeneous propagation. The diffusion process for the first campaign (with the second campaign following the same process) proceeds as follows:
    i. At step $t = 0$, nodes in the seed set $U_1$ are activated, while all the other nodes remain inactive.
    ii. In each step $t$, any active user $u$ for campaign $c_1$ attempts to activate each of its inactive outgoing neighbors $v$ with a probability $p^1_{(u,v)}$. This activation is like flipping a coin with a probability of heads equal to $p^1_{(u,v)}$: if heads, $v$ is activated; if tails, $v$ remains inactive. Each active user $u$ has only one chance to activate each neighbor for campaign $c_1$, after which $u$ stays active but ceases further activation efforts.
    iii. The diffusion process ends when no more nodes can be activated.
- **Exposed Node Set** – $r_i(U)$ represents the set of vertices that can be reached from $U$ through the cascade

process mentioned above for campaign $c_i$, for a given seed set $U$. Note that in a single propagation step, $r_i(U)$ includes not only nodes that were successfully activated by $U$ but also nodes that were targeted for activation by $U$ but remained inactive. Since the diffusion process is stochastic, $r_i(U)$ is a random variable.

### B. *Problem Formulation*

Given a social network $G = (V, E)$, two sets of $I_1$ and $I_2$ of the initial seed sets for the two campaigns, and a budget $k$, the goal of the IEMP is to identify two sets $S_1$ and $S_2$ such that $|S_1| + |S_2| \leq k$ and to maximize the expected number of vertices that are either influenced by both campaigns or remain unaffected by both; that is,

$$\max \Phi(S_1, S_1) = \max \mathbb{E}[|V \setminus (r_1(I_1 \cup S_1) \triangle r_2(I_2 \cup S_2))|]$$

$$\text{such that } |S_1| + |S_2| \leq k \text{ and } S_1, S_2 \in V$$

## III. Methodology

As introduced earlier, IEMP's diffusion model is based on a stochastic process. The two algorithms outlined below will leverage this diffusion model within their respective procedures. To facilitate this, it is essential to define an algorithm that accurately represents the diffusion process, which can be accomplished through the application of a breadth-first search (BFS) approach.

**Algorithm 1: Influence Diffusion Via BFS**

```
Input: graph(G), seed ($U_i = I_i \cup S_i$), campaign (campaign index)
Output: active (set of active nodes), exposed (set of exposed nodes)
1  Initialize queue as an empty queue
2  Initialize active as an empty set
3  Initialize exposed as an empty set
4  for node in seed do
5      queue.Add(node)
6      active.Add(node)
7      exposed.Add(node)
8  while queue is not emoty do
9      node ← queue.Remove()
10     for neighbor, probab in graph.GetAdjList(node) do
11         if not exposed.Contains(neighbor) then
12             exposed.Add(neighbor)
13         if not active.Contains(neighbor) then
14             if probab ≥ RandomValue() then
15                 active.Add(neighbor)
16                 queue.Add(neighbor)
17 return active, exposed
```

### A. *Heuristic Search*

This section outlines a heuristic approach for solving the IEMP using a Monte Carlo-based greedy selection strategy.

The main idea of this heuristic algorithm is to expand the node with the highest incremental influence value, $h(v)$, to maximize exposure for two campaigns.

To achieve a better computational accuracy, the algorithm applies Monte Carlo simulation to approximate each node's influence impact by averaging results from multiple simulated influence spreads. Furthermore, to improve efficiency, the algorithm shall avoid recalculating the entire influence spread from scratch. It should only compute the changes in active and exposed nodes for each candidate node. This is achieved by diffusing influence to only a depth deeper and incrementally updates active and exposed sets.

In this proposed influence diffusion process, each incremental computation is conducted to a depth of one, focusing solely on evaluating the immediate neighbors of the current source node. This approach does not lead to issues with activation or exposure, as the algorithm will subsequently iterate through all nodes, prompting each node to activate its own neighbors. Consequently, there is no need to explore deeper from the initial node, as doing so would introduce unnecessary redundancy in the checks performed.

**Algorithm 2: Incremental Influence Diffusion**

```
Input: graph, src (source node), active, exposed, campaign
Output: active_increment (set of newly activated nodes), exposed_increment (set of newly exposed nodes)
1  Initialize active_increment as an empty set
2  Initialize exposed_increment as an empty set
3  if not active.contains(src) then
4      active_increment.Add(src)
5  if not exposed.contains(src) then
6      exposed_increment.Add(src)
7  for neighbor, probab in graph.GetAdjList(src) do
8      if not exposed.Contains(neighbor) then
9          exposed_increment.Add(neighbor)
10     if not active.Contains(neighbor) then
11         if probab ≥ RandomValue() then
12             active_increment.Add(neighbor)
13 return active_increment, exposed_increment
```

By limiting the depth to one at this stage, we streamline the process and enhance computational efficiency while ensuring comprehensive coverage of the network in subsequent iterations. Nonetheless, the aforementioned influence diffusion with exhaustive BFS is still utilized in our final algorithm, but only once for each campaign.

**Algorithm 3: IEMP Monte Carlo Greedy Heuristic**

```
Input: graph, initial1 ($I_1$), initial2 ($I_2$), budget ($k$), rep (number of simulations)
Output: balanced1 ($S_1$), balanced2 ($S_2$)
1  num_nodes ← graph.GetNumNodes()
2  Initialize balanced1 as an empty set
3  Initialize balanced2 as an empty set
4  while balanced1.Length() + balanced2.Length() < budget do
5      Initialize h1_rec as an empty array
6      Initialize h2_rec as an empty array
```

```
7     for $j \leftarrow 0$ to $rep$-1 do
8         $union1 \leftarrow initial1 \cup balanced1$
9         $union2 \leftarrow initial2 \cup balanced2$
10        $active1, exposed1 \leftarrow$ InfluenceDiffusionViaBFS($graph, union1$, 1)
11        $active2, exposed2 \leftarrow$ InfluenceDiffusionViaBFS($graph, union2$, 2)
12        $phi\_s1\_s2 \leftarrow$ ComputePhi($num\_nodes, exposed1, exposed2$)
13        for $i \leftarrow 0$ to $num\_nodes$-1 do
14            if not $balanced1$.contains($i$) and not $balanced2$.contains($i$) then
15                $active1\_increment, exposed1\_increment \leftarrow$ IncrementalInfluenceDiffusion($graph, i, active1, exposed1$, 1)
16                $active2\_increment, exposed2\_increment \leftarrow$ IncrementalInfluenceDiffusion($graph, i, active2, exposed2$, 2)
17                $phi\_s1vi\_s2 \leftarrow$ ComputePhi($num\_nodes, exposed1 \cup exposed1\_increment, exposed2$)
18                $phi\_s1\_s2vi \leftarrow$ ComputePhi($num\_nodes, exposed1, exposed2 \cup exposed2\_increment$)
19                $h1\_rec[i] \leftarrow h1\_rec[i] + phi\_s1vi\_s2 - phi\_s1\_s2$
20                $h2\_rec[i] \leftarrow h2\_rec[i] + phi\_s1\_s2vi - phi\_s1\_s2$
21    for $j \leftarrow 0$ to $num$_nodes-1 do
22        $h1\_rec[j] \leftarrow h1\_rec[j]$ / $rep$
23        $h2\_rec[j] \leftarrow h2\_rec[j]$ / $rep$
24    $new\_v1 \leftarrow$ IndexOfMaxElement($h1\_rec$)
25    $new\_v2 \leftarrow$ IndexOfMaxElement($h2\_rec$)
26    if $new\_v1 \geq new\_v2$ then
27        $balanced1$.Add($new\_v1$)
28    else
29        $balanced2$.Add($new\_v2$)
30 return $balanced1, balanced2$
```

## B. *Evolutionary Approach*

In this section, we address the IEMP by employing a genetic algorithm (GA) to identify optimal intervention sets $S_1$ and $S_2$ that maximize the expected influence spread within a network. Given the combinatorial complexity and large search space of this problem, optimization techniques like genetic algorithms provide an effective solution method. GAs are well-suited for this task due to their capability to explore complex, multimodal search spaces and to find near-optimal solutions within reasonable computational time. The GA approach is structured as follows:

- **Solution Representation**: Each candidate solution is represented by a binary vector, $x = \{x_1, x_2, \ldots, x_{|V|}, x_{|V+1|}, x_{|V+2|}, \ldots, x_{|V+V|}\}$ where $x_i \in \{0, 1\}$.
- **Fitness Function**: To guide the search towards optimal solutions, a fitness function is defined that evaluates each candidate solution based on the influence spread it achieves. The fitness function distinguishes between feasible and infeasible solutions.

$$fitness(S_1, S_2) = \begin{cases} \Phi(S_1, S_2) \text{ if } |S1| + |S2| \leq k \\ -\Phi(S_1, S_2) \text{ otherwise} \end{cases}$$

- **Genetic Algorithm Operations**: In our approach, diversity is cherished; hence, the GA operations below is full of randomness, which makes this study quite experimental.
   i. **Initialization**: The initial population is generated in three distinct groups to balance diversity and quality: The random population is created with random binary vectors where nodes are randomly added to intervention sets *S1* or *S2*. The controlled random population has nodes selected within a controlled range, increasing the probability of feasible solutions within budget. The ideal population is generated with exactly the budgeted number of nodes, creating solutions that strictly satisfy constraints.
   ii. **Survivor Selection and Offspring Generation**: Survivor selection is relatively simple. It involves choosing the top-performing individuals and a few mid-range and lower-performing solutions to maintain genetic diversity. Choosing pairs for breeding is done through random pair selection, except some controlled pairs for breeding among the best parents. Offspring generation is achieved through two-point crossover and mutation, where selected individuals exchange segments of their binary vectors to produce new solutions. Flip-bit mutation is applied probabilistically to encourage diversity, with a defined mutation rate applied to randomly chosen bits in the binary vectors of offspring solutions. This technique introduces further variation into the population and prevents premature convergence.

**ALGORITHM 4: IEMP GENETIC ALGORITHM**

```
   Input: $graph, initial1, initial2, budget$
   Output: $balanced1, balanced2$
1  $HALT \leftarrow 0.98$
2  $num\_nodes \leftarrow graph$.GetNumNodes()
3  $gen0 \leftarrow$ InitGen0(graph, $budget$)
4  Evaluate fitness of $gen0$
5  Sort $gen0$ based on fitness value
6  $current\_best\_solution, current\_best\_solution\_val \leftarrow$ RetrieveBestSolution($gen0$)
7  $current\_gen \leftarrow gen0$
8  for $generation \leftarrow 0$ to $gen$-2 do
9      if $current\_best\_solution\_val$ / $num\_nodes \geq$ HALT then
10         break
11     $parent\_gen \leftarrow$ ChooseSurvivor($current\_gen$)
12     $next\_gen \leftarrow$ GenerateOffSpring($parent\_gen$)
```

```
13      Evaluate fitness of $next\_gen$
14      Sort $next\_gen$ based on fitness value
15      $candidate\_best\_solution$,
        $candidate\_best\_solution\_val \leftarrow$
        RetrieveBestSolution($next\_gen$)
16      if $candidate\_best\_solution\_val \geq$
        $current\_best\_solution\_val$ then
17          $best\_solution \leftarrow candidate\_best\_solution$
18          $best\_solution\_val \leftarrow$
            $candidate\_best\_solution\_val$
19  $balanced1, balanced2 \leftarrow$
    ConvertBinaryRepresentationToSetRepresentation(
    $best\_solution$)
20  return $balanced1, balanced2$
```

Note that we may terminate the algorithm as soon as the halt condition is satisfied, which is defined as checking if $\Phi(S_1, S_1)$ is 98% as big as the total number of nodes. This allows us to improve the computational time by accepting results that are sufficiently close to optimal.

## IV. Experiments

### A. Setup

In our experiment, we conducted Monte Carlo simulations three times within our heuristic algorithm for each test case to account for stochastic variability and improve result reliability. For the evolutionary algorithm, we configured a population size of 90 and limited the evolutionary generations to 50, balancing computation time with solution quality. Additionally, we ran the evolutionary algorithm both with and without the halt condition to explore how closely it could approach optimal results with and without this constraint.

We tested our approach on three datasets of varying scale and structure, as shown in Table I. Dataset 1 provided a baseline with a smaller network size, while Dataset 2 introduced a moderate level of complexity. Dataset 3, the largest, presented additional challenges due to its scale and influence probabilities skewed towards lower values, which could limit spread and increase unpredictability. For each dataset, we ran the tests three times and computed the average of the final evaluation values based on the outputted balanced seed sets for a better accuracy in performance comparison.

TABLE I. Datasets' Details

| Dataset | Number of Nodes | Number of Edges |
|---|---|---|
| 1 | 475 | 13289 |
| 2 | 13984 | 17319 |
| 3 | 36742 | 49248 |

The testing environment for this experiment was configured on macOS Sequoia version 15.0.1, running on an Apple M3 Pro processor with 18GB of RAM and a 1TB storage capacity. The programming language used for the algorithm implementation was Python 3.12, with essential libraries such as Numpy 2.1 for handling numerical computations.

### B. Results

To obtain evaluation values, we ran independent cascade to simulate the influence spread on each outputted balanced seed set for 300 times. We also recorded the running time in the process.

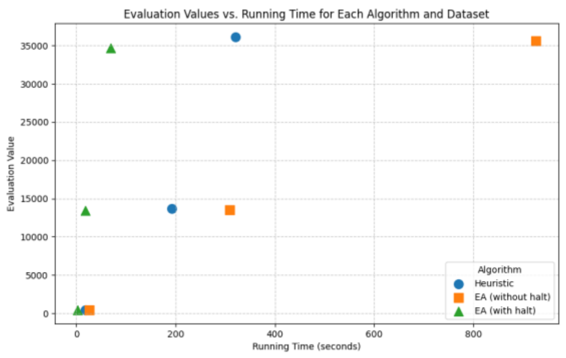

Fig. 1. Evaluation values and running time for each algorithm across different datasets

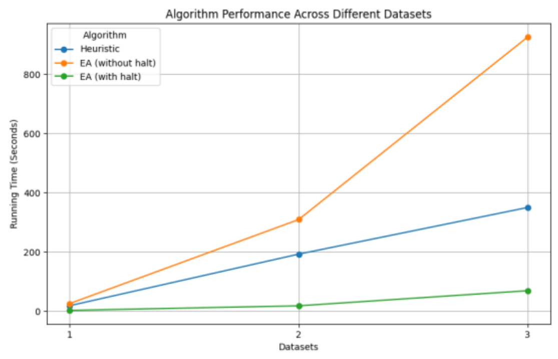

Fig. 2. Running time across different datasets

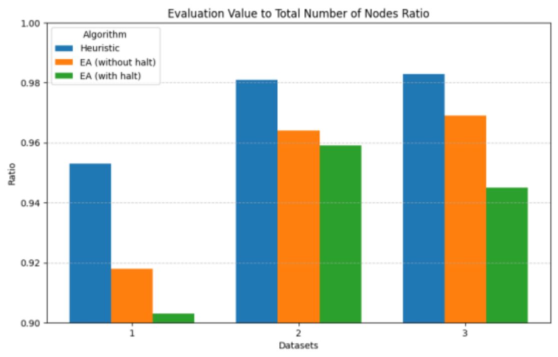

Fig. 3. Evaluation value to total number of nodes ratio across different datasets

In the halt condition of the evolutionary algorithm, we only ran independent cascade 10 times for each instance, which resulted in a high but less accurate evaluation value. This allowed the halt condition to satisfy and terminate the algorithm early, even though the actual ratio upon running 300 simulations is not that large as shown in Figure 3.

### C. Analysis

In analyzing the results presented in the grouped bar chart (Figure 3), we observe that the Heuristic method consistently outperforms both Evolutionary Algorithms (EA) across all datasets in terms of the evaluation value to total number of nodes ratio. Specifically, the Heuristic achieves ratios of 0.953, 0.981, and 0.983 for Datasets 1, 2, and 3, respectively. In comparison, the EA without halt shows a slight drop in performance, with ratios of 0.918, 0.964, and 0.969, while the EA with halt presents the lowest ratios, indicating its relatively diminished efficiency with values of 0.903, 0.959, and 0.945. These results suggest that while all methods demonstrate high effectiveness, the Heuristic method is notably superior, especially in scenarios demanding optimal results.

The running time performance depicted in Figure 2 reveals a stark contrast between the Heuristic method and the EAs. While the Heuristic maintains a significantly lower running time across all datasets—recording 18.11 seconds for Dataset 1, 192.35 seconds for Dataset 2, and 350.03 seconds for Dataset 3—the EAs exhibit a marked increase in time consumption, particularly the EA without halt, which takes 25.27 seconds for Dataset 1 and skyrockets to 925.81 seconds for Dataset 3. In contrast, the EA with halt presents the most favorable time performance for larger datasets, showing a drastic reduction in running time, down to 68.82 seconds for Dataset 3. This indicates that while the Heuristic method excels in efficiency, the halting mechanism of the EA may optimize performance and be able to achieve a close-to-optimal results in significantly less time.

Finally, the scatter plot presented in Figure 1 provides a comprehensive visualization of the relationship between evaluation values and running times across algorithms. It illustrates that the Heuristic consistently achieves high evaluation values with relatively low running times. Notably, the data points for the Heuristic cluster towards the higher evaluation values, contrasting sharply with the EAs, which, despite having longer runtimes, exhibit a less consistent performance in evaluation values. The EA with halt, although showing improvements in runtime, still fails to reach the evaluation levels of the Heuristic. This analysis emphasizes that while running time is an essential factor in algorithm selection, the balance between efficiency and effectiveness is crucial for achieving optimal results in practical applications, which makes Heuristic our overall best approach.

## V. Conclusion

In this study, the heuristic and evolutionary algorithms were applied to the IEMP to maximize information exposure within a social network. The heuristic method consistently showed superior performance in terms of efficiency and effectiveness across all datasets. It achieved higher evaluation values and required significantly less computational time compared to the evolutionary algorithms. However, the evolutionary algorithm with a halt condition provided a reasonable compromise by reducing runtime without severely compromising results, making it potentially suitable for larger networks where time constraints are critical.

While the heuristic approach proves robust, further exploration into other evolutionary algorithms, such as particle swarm optimization or differential evolution, may provide improved solutions. Additionally, hybrid models that combine heuristic elements with genetic algorithms could offer a balance between accuracy and computation time. This study demonstrates the potential of heuristic algorithms in tackling complex optimization problems within influence maximization, with the evolutionary approach presenting an adaptable framework for broader applications in network analysis.

## References

[1] K Garimella, A Gionis, N Parotsidis, N Tatti. Balancing information exposure in social networks. NeurIPS 2017: 4663-4671.