Multi-Hop Influence Maximization in Social Networks

Francisco Richter

1 Introduction

The goal of the Multi-Hop Influence Maximization problem, also known as the k-d-Dominating Set Problem (k-dDSP), is to identify a set of influential nodes in a social network such that the spread of influence is maximized within a certain number of hops (steps). This problem can be represented using a directed graph $G = (V, A)$, where:

- V is the set of nodes (representing individuals in the network).
- A is the set of directed arcs (representing relationships or influence paths between individuals).

2 Key Terms

2.1 Influence $I_d(u)$

This represents the set of nodes that can be reached from a node u within d hops. The parameter d is the maximum distance (in terms of hops or steps) that influence can spread from a given node.

2.2 Distance Measure $dist(u, v)$

This is defined as the length (number of arcs) of the shortest directed path from node u to node v.

3 Formal Definition

For a node $u \in V$, the influence set $I_d(u)$ is defined as:

$$I_d(u) = \{v \in V \mid dist(u, v) \leq d\}$$

This means $I_d(u)$ includes all nodes v that can be reached from u within d hops.

For a set of nodes $U \subseteq V$, the combined influence set $I_d(U)$ is defined as:

$$I_d(U) = \bigcup_{u \in U} I_d(u)$$

This means $I_d(U)$ includes all nodes influenced by any node in the set U.

4 Optimization Objective

The objective of the k-dDSP is to find a set $U^* \subseteq V$ of at most k nodes such that the combined influence set $I_d(U^*)$ is as large as possible. Formally, this can be written as:

$$\max_{U \subseteq V} |I_d(U)|$$

subject to:

$$|U| \leq k$$

In simpler terms, we want to select k nodes that can influence the maximum number of other nodes within d hops.

5 Metaheuristic Approach

In this section, we describe the previous methods used for solving the Multi-Hop Influence Maximization problem: Genetic Algorithm (GA), Simulated Annealing (SA), Lazy Greedy Algorithm (LGA), and Biased Random Key Genetic Algorithm (BRKGA).

5.1 Genetic Algorithm (GA)

The Genetic Algorithm (GA) is a metaheuristic inspired by the process of natural selection. It uses techniques such as selection, crossover, and mutation to evolve a population of candidate solutions towards better solutions. The steps involved in GA are as follows:

1. **Initialization:** Generate an initial population of candidate solutions randomly.
2. **Fitness Evaluation:** Calculate the fitness of each candidate solution.
3. **Selection:** Select a subset of the population based on their fitness to create offspring.
4. **Crossover:** Combine pairs of selected candidates to produce new offspring (children).
5. **Mutation:** Apply random changes to some of the offspring to maintain genetic diversity.

6. **Replacement:** Replace the old population with the new offspring.

7. **Termination:** Repeat the process until a stopping criterion is met (e.g., a fixed number of generations).

5.2 Simulated Annealing (SA)

Simulated Annealing (SA) is a probabilistic optimization technique inspired by the annealing process in metallurgy. It explores the solution space by probabilistically accepting worse solutions to escape local optima. The steps involved in SA are as follows:

1. **Initialization:** Start with an initial solution and an initial temperature.

2. **Neighbor Selection:** Generate a neighboring solution by making a small change to the current solution.

3. **Acceptance Probability:** Calculate the acceptance probability based on the difference in fitness between the current and neighboring solutions and the current temperature.

4. **Acceptance Decision:** Accept the neighboring solution with a probability given by the acceptance probability.

5. **Temperature Update:** Decrease the temperature according to a cooling schedule.

6. **Termination:** Repeat the process until the temperature reaches a minimum value or a stopping criterion is met.

5.3 Lazy Greedy Algorithm (LGA)

The Lazy Greedy Algorithm (LGA) is an optimization technique that leverages the submodular property of the influence maximization problem to efficiently find a near-optimal solution. The steps involved in LGA are as follows:

1. **Initialization:** Initialize an empty set of selected nodes.

2. **Priority Queue:** Use a priority queue to keep track of the marginal gain of each node.

3. **Node Selection:** Iteratively select the node with the highest marginal gain and update the priority queue.

4. **Marginal Gain Update:** After selecting a node, update the marginal gain of the remaining nodes in the priority queue.

5. **Termination:** Repeat the process until the desired number of nodes is selected.

5.4 Biased Random Key Genetic Algorithm (BRKGA)

A BRKGA is a type of genetic algorithm where individuals (solutions) are represented as vectors of real numbers (random keys). The algorithm involves generating an initial population, selecting the best solutions, creating new solutions through crossover and mutation, and repeating this process to improve the solutions.

Algorithm 1 The pseudocode of BRKGA

Require: a directed graph $G = (V, E)$
Ensure: values for parameters $psize$, pe, pm, $probelite$, $seed$
  $P \leftarrow$ GenerateInitialPopulation($psize, seed$)
  Evaluate(P) {problem-dependent part (greedy)}
  while computation time limit not reached **do**
    $Pe \leftarrow$ EliteSolutions(P, pe)
    $Pm \leftarrow$ Mutants(P, pm)
    $Pc \leftarrow$ Crossover($P, pe, probelite$)
    Evaluate(Pm $\cup Pc$)$\{problem - dependentpart(greedy)\}P \leftarrow Pe \cup Pm \cup Pc$
  end while
  return Best solution in P

The algorithm begins by generating an initial population P of $psize$ individuals. These individuals are evaluated by transforming each individual into a valid solution to the k-dDSP. The evaluation function uses a greedy heuristic based on the out-degree of nodes. The final solution is obtained by selecting the k nodes with the highest greedy values.

6 Challenges

The problem is NP-hard, meaning that finding the exact optimal solution is computationally infeasible for large networks. Heuristic and metaheuristic approaches are used to find good-enough solutions in reasonable time.

7 Comparative Analysis

In this section, we present a comparative analysis of the performance and efficiency of the heuristic methods discussed: Genetic Algorithm (GA), Simulated Annealing (SA), Lazy Greedy Algorithm (LGA), and Biased Random Key Genetic Algorithm (BRKGA). The evaluation is based on two primary metrics: computational efficiency and the quality of the solutions obtained.

7.1 Performance Metrics

To evaluate the performance of the heuristic algorithms, we used two key metrics:

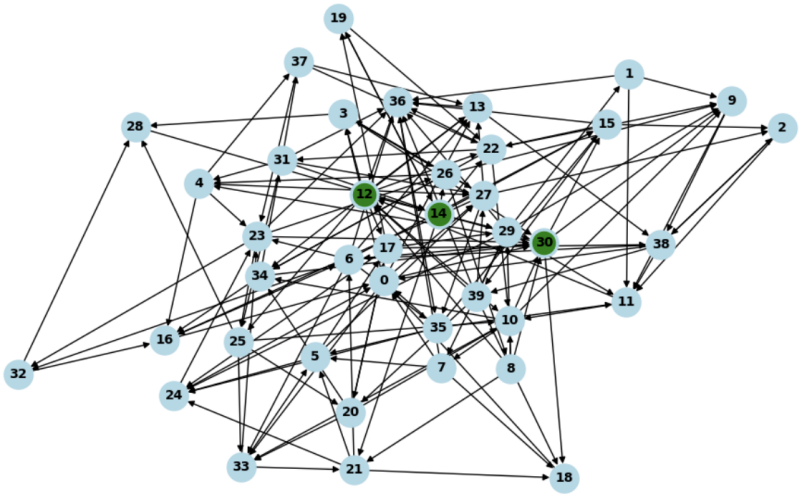

Figure 1: Example of a multi-hop influence process in a directed graph with 40 nodes. The selected nodes (in green) maximize influence within 1-hop, 2-hop, and 3-hop distances.

- **Computational Efficiency**: This metric measures the time taken by each algorithm to execute and converge to a solution. It is crucial for understanding the practicality of deploying these algorithms on large-scale networks.
- **Quality of Solutions**: This metric assesses the effectiveness of each algorithm in maximizing influence, represented by the number of influenced nodes in the network.

7.2 Computational Efficiency

Figure 2 illustrates the computational efficiency of the algorithms as a function of the number of nodes, fixed at 100 nodes. The graph indicates that the Lazy Greedy Algorithm (LGA) consistently outperforms the other methods in terms of computational time, demonstrating its suitability for larger networks where efficiency is paramount. The Genetic Algorithm (GA) and Biased Random Key Genetic Algorithm (BRKGA) show competitive performance, with BRKGA slightly faster due to its more refined selection and crossover mechanisms. Simulated Annealing (SA), while effective, tends to be slower, particularly as the network size increases, due to its iterative nature and temperature scheduling.

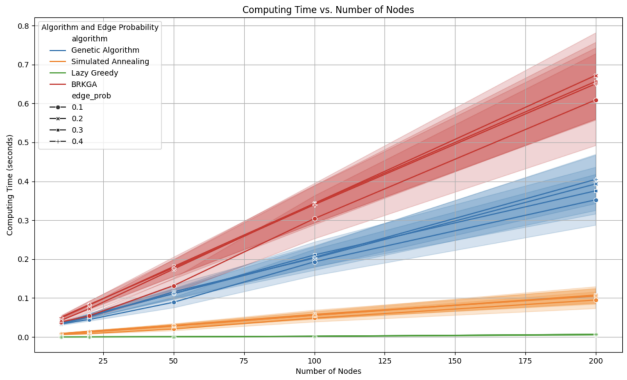

Figure 2: Computing Time vs. Number of Nodes (Fixed 100 Nodes)

7.3 Quality of Solutions

The quality of the solutions produced by each algorithm, measured by the number of influenced nodes, is shown in Figure 3. This graph focuses on a fixed edge probability to provide a clear comparison. The Lazy Greedy Algorithm (LGA) consistently achieves the highest influence spread, leveraging its submodular property to identify highly influential nodes efficiently. The Genetic Algorithm (GA) and BRKGA also perform well, with BRKGA showing a slight edge due to its biasing mechanism which helps in maintaining diverse yet promising solutions. Simulated Annealing (SA), while effective, tends to achieve slightly lower influence spread compared to the other methods.

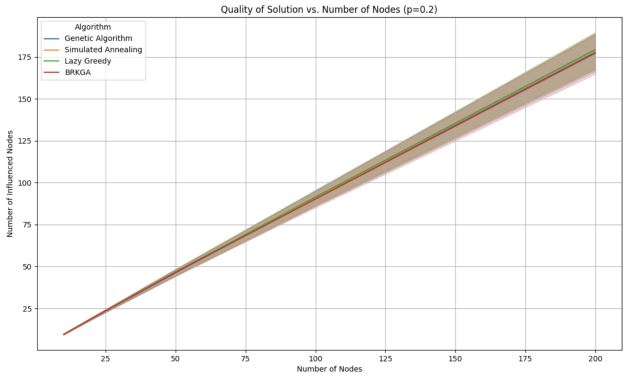

Figure 3: Quality of Solution vs. Number of Nodes (Fixed Edge Probability)

7.4 Impact of the Number of Influential Nodes (k)

The effect of varying the number of influential nodes (k) on the computational efficiency and quality of solutions is depicted in Figures 4 and 5, respectively. The computational time generally increases with k, reflecting the additional complexity of selecting a larger set of influential nodes. LGA remains the most efficient, while SA shows the highest increase in computational time.

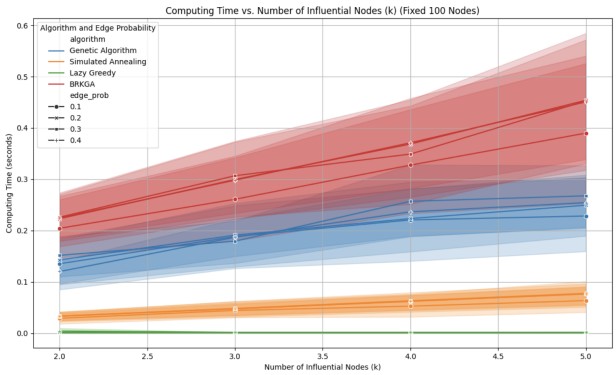

Figure 4: Computing Time vs. Number of Influential Nodes (k) (Fixed 100 Nodes)

As k increases, the quality of the solutions, as shown in Figure 5, improves for all algorithms, with LGA again leading, followed by BRKGA and GA. SA, while still effective, lags behind the others in terms of the influence spread achieved.

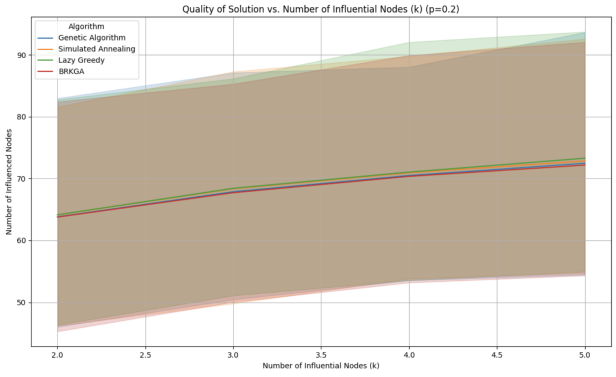

Figure 5: Quality of Solution vs. Number of Influential Nodes (k) (Fixed Edge Probability)

7.5 Effect of Edge Probability

Figures 6 and 7 explore the influence of varying edge probabilities on computational time and solution quality, with the number of nodes fixed at 100. Higher edge probabilities typically increase computational complexity due to denser graphs, which is evident in the increased computing times for all algorithms. LGA remains the most efficient, while BRKGA and GA show moderate increases in computation time.

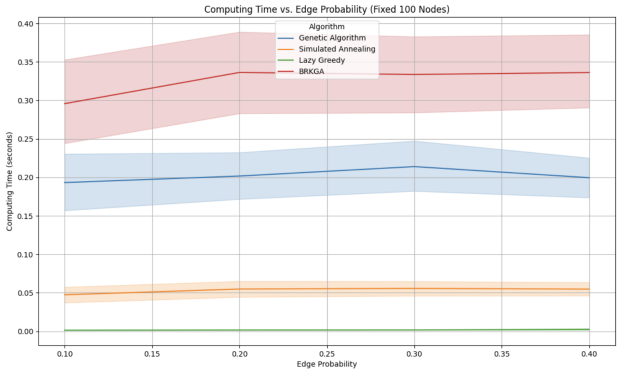

Figure 6: Computing Time vs. Edge Probability (Fixed 100 Nodes)

In terms of solution quality (Figure 7), higher edge probabilities enhance the influence spread capabilities of all algorithms. LGA continues to outperform, followed by BRKGA and GA, with SA trailing behind.

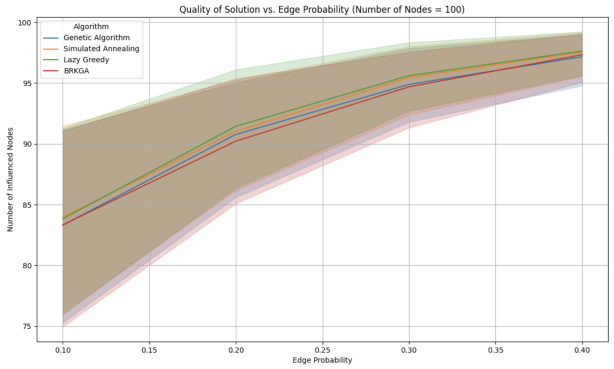

Figure 7: Quality of Solution vs. Edge Probability (Number of Nodes = 100)