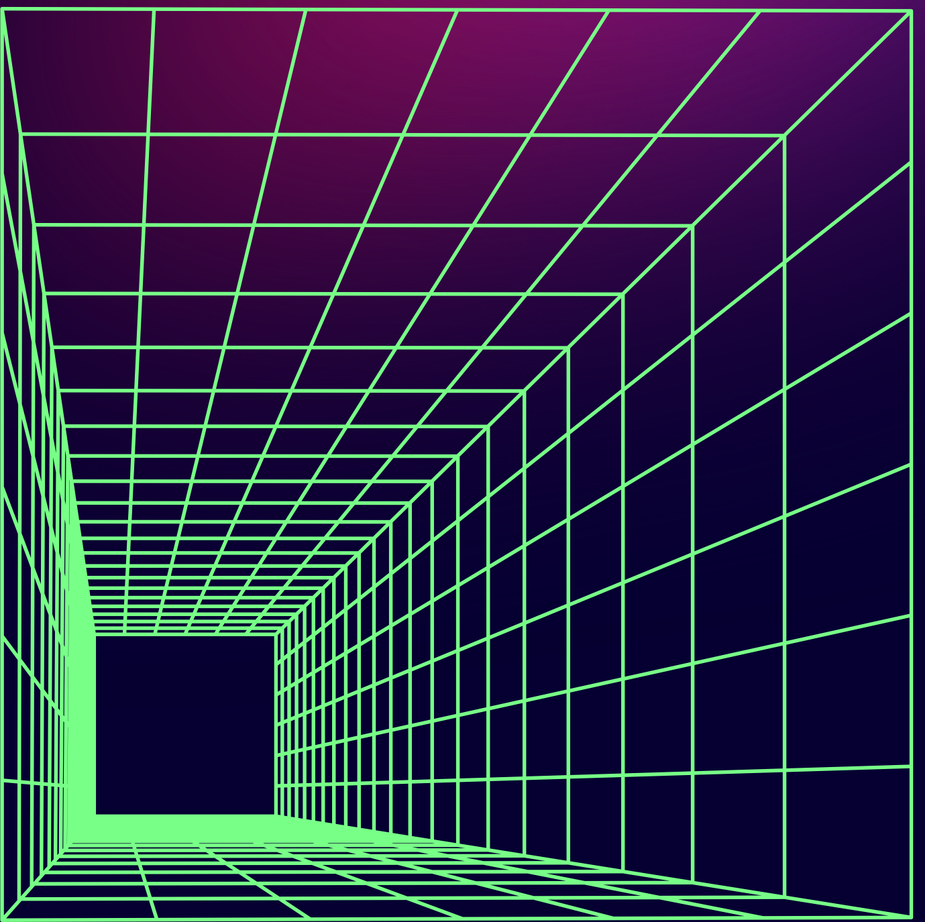
INFLUENCE MAXIMIZATION

AADI AGRAWAL
ADITYA ANAND

INTRODUCTION



Social networks have long been a nature of the humanity life, where distinguished individuals of a community could drive members of their community into embracing a faith or adopting a behaviour or change their life conditions .

This natural humanity trait, has subsequently been digitalised in Online Social Networks.

The propagation pace in OSNs is much faster than in real-life networks, and connections between OSN users scale much higher and quicker in OSNs.

OSNs are investigated using concepts from graph theory in the context of computer science and social science.
A graph representation of OSNs include nodes or vertices that represent OSN individuals, and links or edges that represent social ties, such as friendship.

GENERAL APPROACH

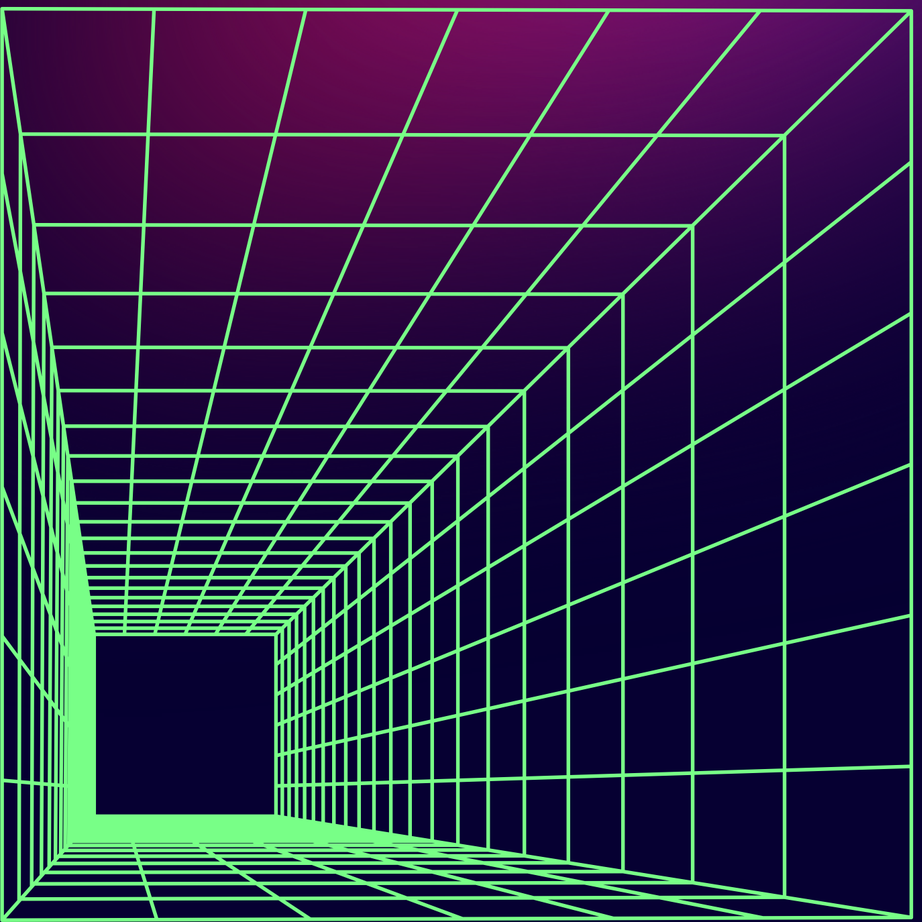
We select a set of k nodes in a social network that are discovered based on both their structural and historical-actions attributes to maximize influence prop agation.

- Initially, “artificial” communities are built whereby “similar” users are assembled together, based on a judicious similarity function.

- Next, for each of these synthetic communities, we identify key users using a computational intelligence technique that employs a fuzzy-logic based function to discriminate nodes based on both their structural-centrality and influence-weight attributes.

- Finally, we rank these key users based on their influ ence power, by simulating an influence diffusion process to determine the seed-set of candidate influence propagation nodes.

GENERAL APPROACH



The final step in the above three-steps process predicts the most influential members from candidate key nodes based on the available marketing budget to shortlist agents for promoting products or services and recommend ing them around social networks.

PROBLEM DESCRIPTION

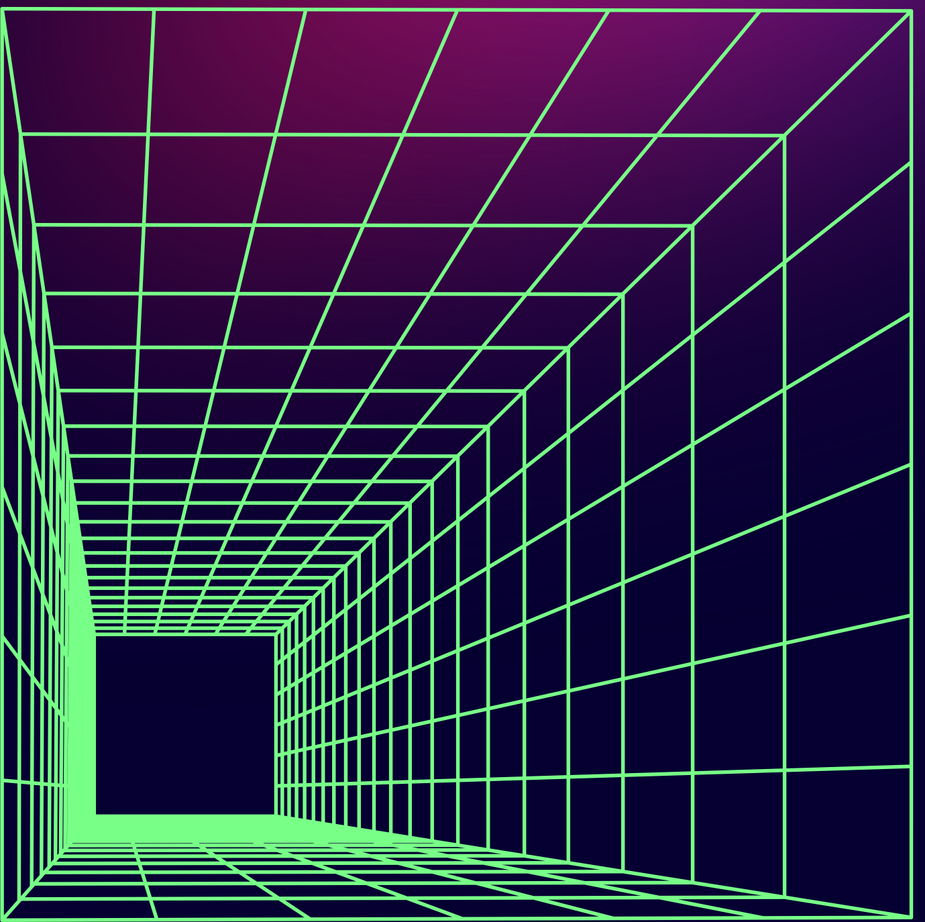
To model members and inherent connections, a graph G (N, E), represents the set of members N ={1,2,...,n} which is implemented as an n ×n adjacency matrix.
The influence weights: 0 ⩽ Eij ⩽ 1 are the adjacency matrix entries.

An influence occurs when a user u of a social network represented by the graph G embraces a behaviour, that was previously embraced by another user v, in which case v is said to have activated u or u is said to have been activated.

Subsequently, influence maximization consists in discovering a subset of key users U in the social network modelled by the graph G, where U= k .

Identifying U is the core problem of influence maximization.

PROBLEM DESCRIPTION



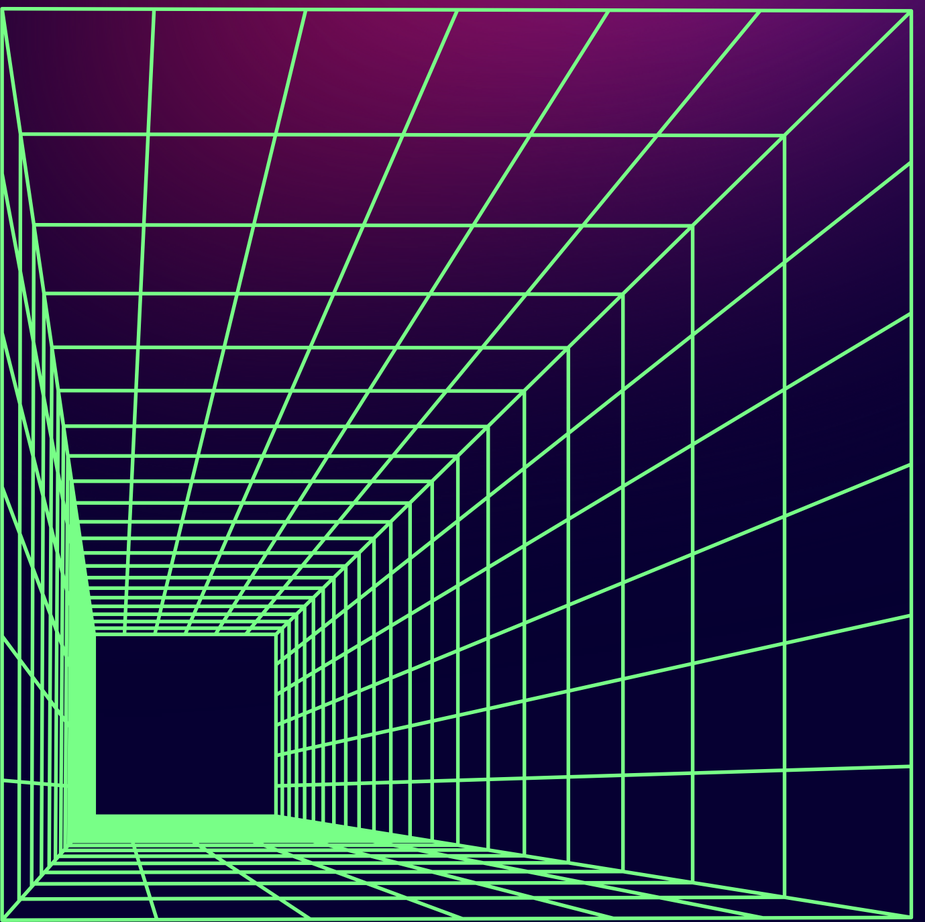
The discovery of key users is subject to a multi-criteria decision-making dilemma due to the contribution of both nodes' influence weights and their topological attributes within the graph, thus this problem motivates to incorporate fuzzy-logic based model.

This theory tolerates logical assertions to carry a progressive extent of values that lie within the interval [0, 1] as an alternative to true/false assertions.

It appears more natural to mimic human, rather than machine reasoning to evaluate real world considerations.

The joint criteria used to determine "key nodes" membership is found to be effectively addressed using a fuzzy-logic based membership function

ILLUSTRATIVE SCENARIO INCORPORATING FUZZY

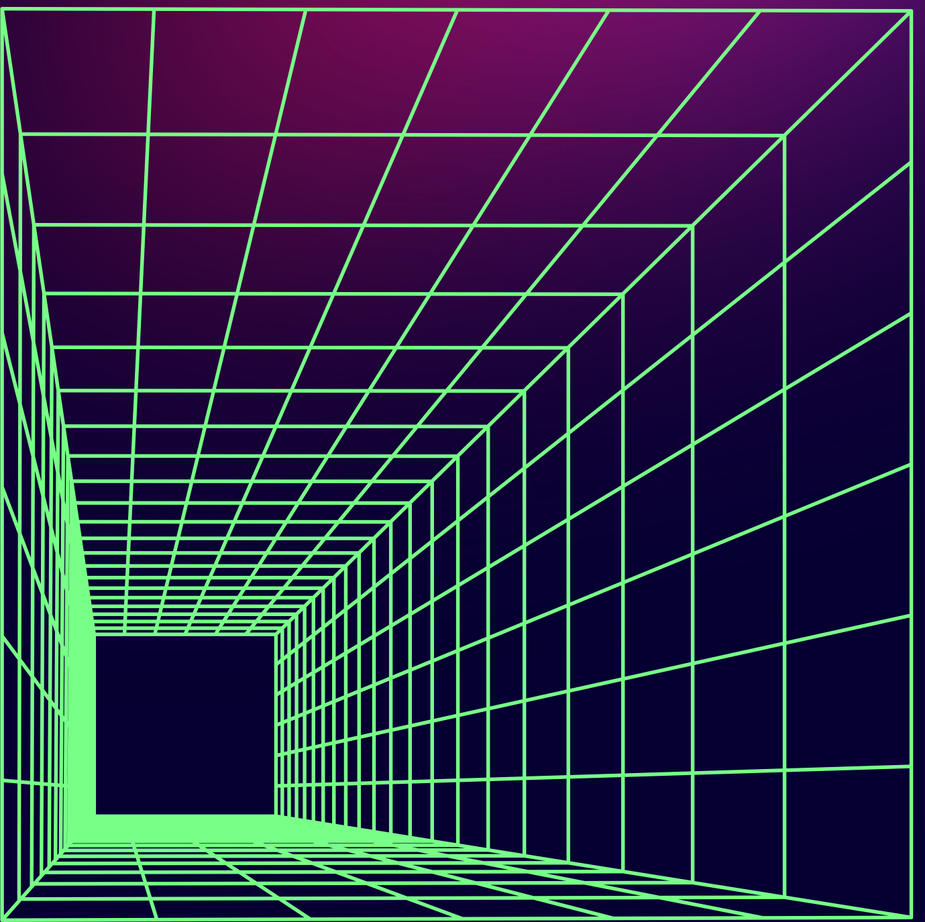


Consider the problem of discovering the most influential users in a social network made up of the following nodes ={1,2,3,4,5} . The goal is to determine nodes that combine centrality location and influence-weight on other nodes attributes.

Centrality (C)={{1,0.4},{2,0.6},{3,0.8},{4,0.4},{5,0.5}}

InfluenceWeight(IW)={{1,0.1},{2,0.9},{3,0.7},{4,1}, {5, 0.2}} represents the influence power of each node.

ILLUSTRATIVE SCENARIO INCORPORATING FUZZY



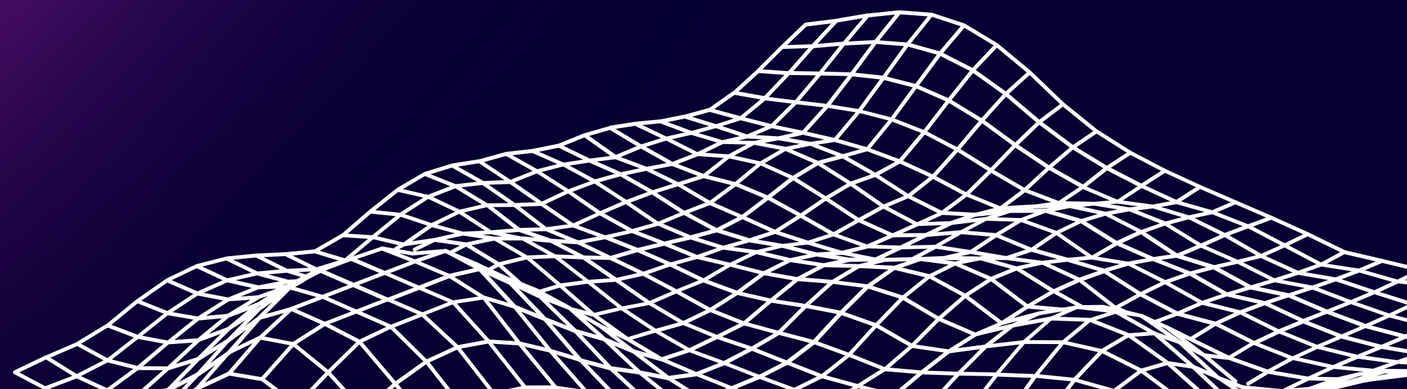
Next, we engage into a decision-making process to identify the set of nodes that are most influential considering both of the above constraints, i.e. favourable location vs. influence weight.

We take Fuzzy intersection as the join criteria.

Result: $C \cap IW = \{\{1,0.1\},\{2,0.6\},\{3,0.7\},\{4,0.4\},\{5,0.2\}\}$.

JAYA ALGORITHM

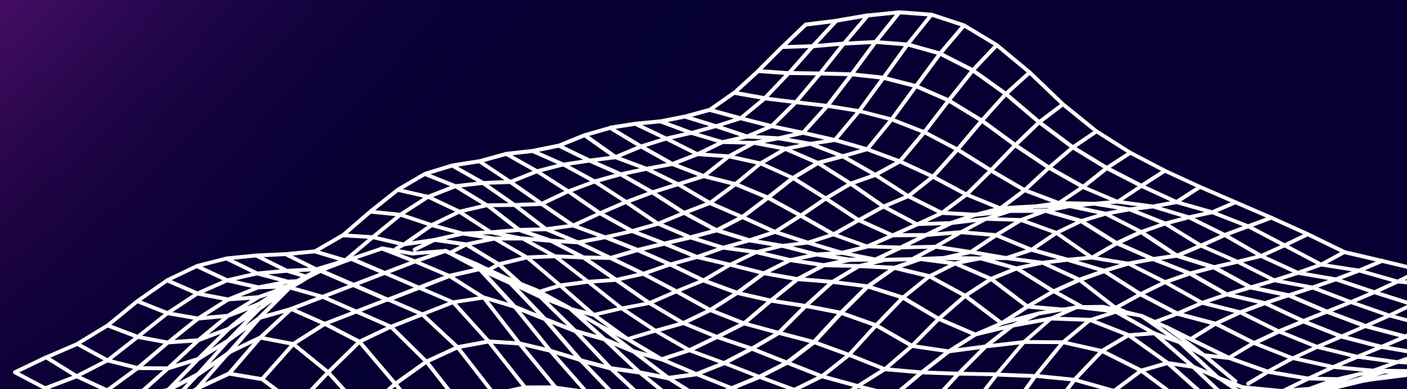
It's a powerful tool belonging to the "swarm intelligence" family, inspired by the "survival of the fittest" principle observed in nature.



JAYA ALGORITHM

What it does:

- Jaya tackles optimization problems, aiming to find the best possible solution for a given objective function. This can involve maximizing profits, minimizing costs, or achieving optimal design parameters.
- It works effectively for both constrained and unconstrained problems, making it versatile for various applications.



JAYA ALGORITHM

How it works:

- Jaya maintains a population of candidate solutions. Each solution represents a potential answer to the optimization problem.
  - Through iterations, the algorithm improves these solutions by:
  - Attracting them towards the best (victorious) solution in the population.
  - Repelling them away from the worst solution, preventing them from getting stuck in local optima.
- This movement is guided by simple mathematical equations involving random numbers, ensuring a balance between exploration (searching for new areas) and exploitation (refining promising solutions).

JAYA ALGORITHM

Mathematical Equations

he Jaya Algorithm relies on simple mathematical principles, it doesn't involve a large number of complex equations.

1. Position Update Equation:

$$X_new = X_current + r1 * (X_best - X_worst) + r2 * (rand() * (X_i2 - X_j2))$$

- X_new: New position of the solution
- X_current: Current position of the solution
- r1, r2: Random numbers in the range [0, 1]
- X_best: Best solution in the population
- X_worst: Worst solution in the population
- X_i2, X_j2: Positions of two randomly chosen different solutions (excluding X_best and X_worst)

JAYA ALGORITHM

Mathematical Equations

Explaination:

- Attraction towards the best: The term r1 * (X_best - X_current) pulls the solution closer to the best solution found so far.
- **Repulsion away from the worst:** The term r2 * (rand() * (X_i2 - X_j2)) pushes the solution away from the worst solution.
- Randomization: The presence of r1, r2, and rand() introduces randomness into the movement. This prevents the algorithm from becoming deterministic and getting stuck in repetitive patterns.

By combining these influences, the position update equation guides each solution towards areas with higher potential for finding the optimal solution. The interplay between attraction, repulsion, and randomness creates a dynamic search process that helps the algorithm converge efficiently.

JAYA ALGORITHM

Mathematical Equations

CONSTRAINT HANDLING

1. Projection:

This method directly projects solutions violating constraints onto the feasible region.

Equation:

X_projected = min(max(X_new, lower_bound), upper_bound)

- X_projected: Solution after projection
- X_new: Newly generated solution
- lower_bound, upper_bound: Limits of the feasible region for each variable

JAYA ALGORITHM

Mathematical Equations

Explanation:

- This equation checks if any element in X_new violates the defined bounds (lower_bound and upper_bound).
- If an element is outside the bounds, it's adjusted to the closest legal value (either the lower or upper bound).
- This ensures all solutions remain within the feasible region.

JAYA ALGORITHM

Flow Chart

Initial Setup:

1. Define your social network: Specify nodes (users), connections, and group memberships (optional).
2. Define your influence function: This determines how likely a user is to be influenced by a neighbor. Options include linear threshold, independent cascade, etc.
3. Set algorithm parameters: Population size, number of iterations, stopping criteria, fuzziness parameter (a).
4. **Define group fairness metric:** Choose a metric to measure fairness across groups (e.g., difference in influence, Gini coefficient).

JAYA ALGORITHM

Flow Chart

Main Loop:

1. **Generate initial population:** Create a set of candidate seed sets. Randomly choose nodes from the network, considering group representation with respect to fairness metric.
    - **Evaluate fitness:** Calculate expected number of influenced users using the influence function.
    - Calculate group fairness score using the chosen metric.
    - Combine fitness and fairness scores into a single objective function using fuzzy logic (if applicable).
2. **Identify best and worst solution:** Find the seed set with the highest combined score (considering both fitness and fairness).

JAYA ALGORITHM

Flow Chart

Main Loop:

1. **Generate initial population:** Create a set of candidate seed sets. Randomly choose nodes from the network, considering group representation with respect to fairness metric.
2. **Evaluate fitness:** Calculate expected number of influenced users using the influence function.
    - Calculate group fairness score using the chosen metric.
    - Combine fitness and fairness scores into a single objective function using fuzzy logic (if applicable).
3. **Identify best and worst solution:** Find the seed set with the highest combined score (considering both fitness and fairness).

JAYA ALGORITHM

Flow Chart

Main Loop:

4. **Update solutions:** For each seed set (except best and worst):Choose two random different seed sets (excluding best and worst).
    - Calculate the difference between the current seed set and both randomly chosen seed sets.
    - **Incorporate Fuzziness (if applicable):** Apply fuzzy operators (e.g., AND, OR) on differences based on their group membership and the fuzziness parameter (α). This introduces flexibility in solution updates.
    - Modify the current seed set by adding a scaled difference towards the best solution and subtracting a scaled difference from the worst solution. Use random numbers for scaling.
    - Ensure updated seed set remains within valid boundaries and maintains diverse group representation.

JAYA ALGORITHM

Flow Chart

Main Loop:

5. Evaluate updated fitness and fairness: Calculate combined score for each updated seed set.

6. Replace worst with better: If any updated solution has a better combined score than the current worst solution, replace the worst with the better one.

7. Check termination: If the stopping criteria are met, proceed to Step 8. Otherwise, go back to Step 2.

-

JAYA ALGORITHM

Flow Chart

Final Output:

The seed set with the highest final combined score represents the set of users with the maximum potential to influence the network while considering group fairness.

JAYA ALGORITHM

Flow Chart

Highlights:

Group Fairness:

- Initial Population: When generating the initial set of candidate seed sets, the algorithm actively considers group representation. This ensures diverse groups are present from the beginning
- **Fitness Evaluation:** Instead of solely relying on the expected number of influenced users, the algorithm introduces a **combined score** that factors in both **influence** and **fairness**.
- Group Fairness Metric: The specific metric used to measure fairness is chosen beforehand and continuously monitored throughout the process. This helps ensure that fairness is not compromised at any stage of the optimization.

JAYA ALGORITHM

Flow Chart

Highlights:

Fuzziness:

- Solution Update: This is where Fuzziness can be applied. During the update process, where differences between solutions are used to guide movement, fuzzy operators can be introduced based on group membership and a fuzziness parameter (a)
- **Fuzzy Operators:** These operators, like AND or OR, introduce **flexibility** in how differences are incorporated.

JAYA ALGORITHM

ALGORITHM

Inputs:

- **Social network:** Graph representing nodes (users) and their connections.
- Influence function: Function defining influence propagation (e.g., linear threshold, independent cascade).
- **Group information:** Optional, defining user group memberships for fairness consideration.
- **Fairness metric:** Function measuring fairness across groups (e.g., Gini coefficient, difference in influence).
- **Fuzziness parameter (α):** Value controlling the degree of fuzziness (optional).
- **Algorithm parameters:** Population size, number of iterations, stopping criteria.

JAYA ALGORITHM

ALGORITHM

Steps:

1. **Function** GenerateInitialPopulation(population_size, α):
   - Randomly select nodes from the network, ensuring diverse group representation based on fairness metric.
     - For each seed set:Calculate expected number of influenced users using the influence function.
     - Calculate group fairness score using the chosen metric.
     - Combine scores into a single objective using fuzzy logic (if applicable).
   - Return the initial population.

JAYA ALGORITHM

ALGORITHM

Steps:

2. **Function** EvaluateSolution(seed_set):
   - Calculate expected number of influenced users using the influence function.
   - Calculate group fairness score using the chosen metric.
   - Combine scores into a single objective using fuzzy logic (if applicable).
   - Return the combined score.

JAYA ALGORITHM

ALGORITHM

Steps:

3. **Main Loop**

- population = GenerateInitialPopulation(population_size, α)
- best_solution, worst_solution = None, None
- For iteration in range(iterations):
  - For solution in population:If solution is not best_solution and solution is not worst_solution:
    - Choose two random different solutions (solution1, solution2) excluding best_solution and worst_solution.
    - Calculate difference vectors: diff1 = solution - solution1, diff2 = solution - solution2.
    - Apply fuzzy operators (based on group membership and α) on diff1 and diff2 (if applicable).
    - Update solution by adding a scaled diff1 towards best_solution and subtracting a scaled diff2 from worst_solution. Use random numbers for scaling.
    - Ensure updated solution remains within valid boundaries and maintains diverse group representation.
    - Evaluate updated solution using EvaluateSolution().
  - Update best_solution and worst_solution based on evaluated scores.
  - Check stopping criteria (e.g., maximum iterations reached, minimal score change).
- Return best_solution.

JAYA ALGORITHM

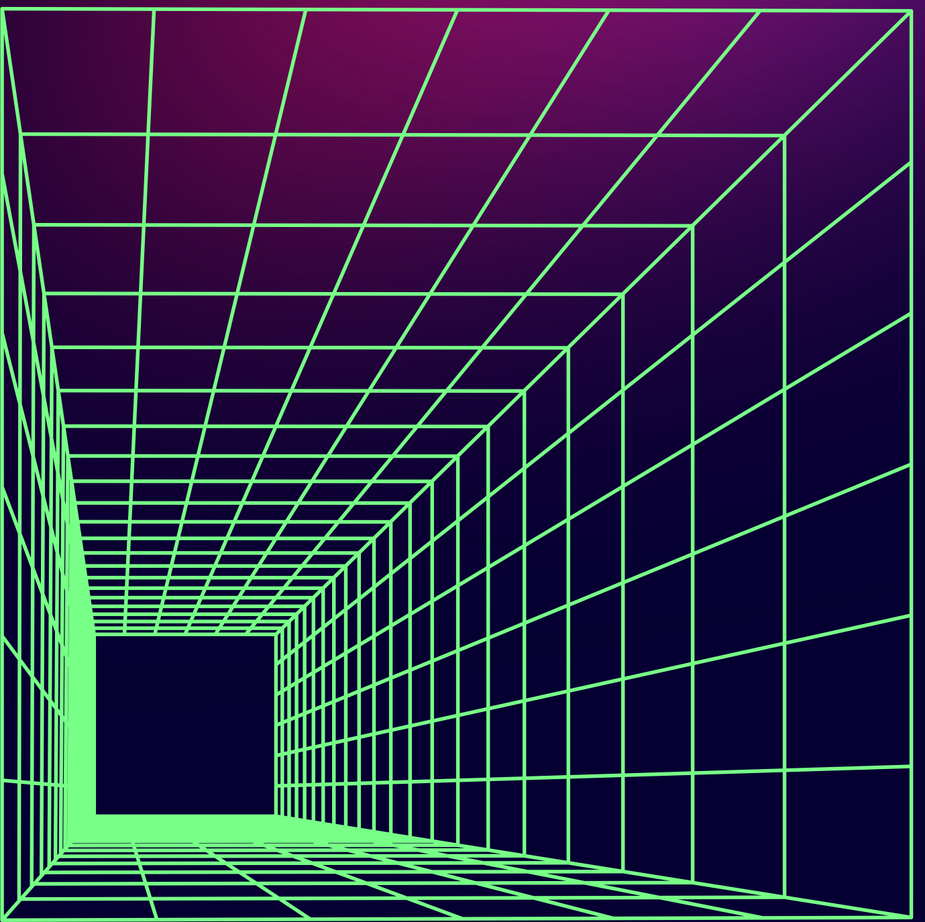
ALGORITHM

Outputs:

Seed set: Set of users maximizing influence while considering fairness.

THANK YOU!

OUTBREAK DETECTION



Similarly, ee explore the general problem of detecting outbreaks in networks, where we are given a network and a dynamic process spreading over this network, and we want to select a set of nodes to detect the process as effectively as possible.

Several optimization criteria to optimize in outbreak detetction such as:

- Minimum detection Time
- Minimum popoulation affected

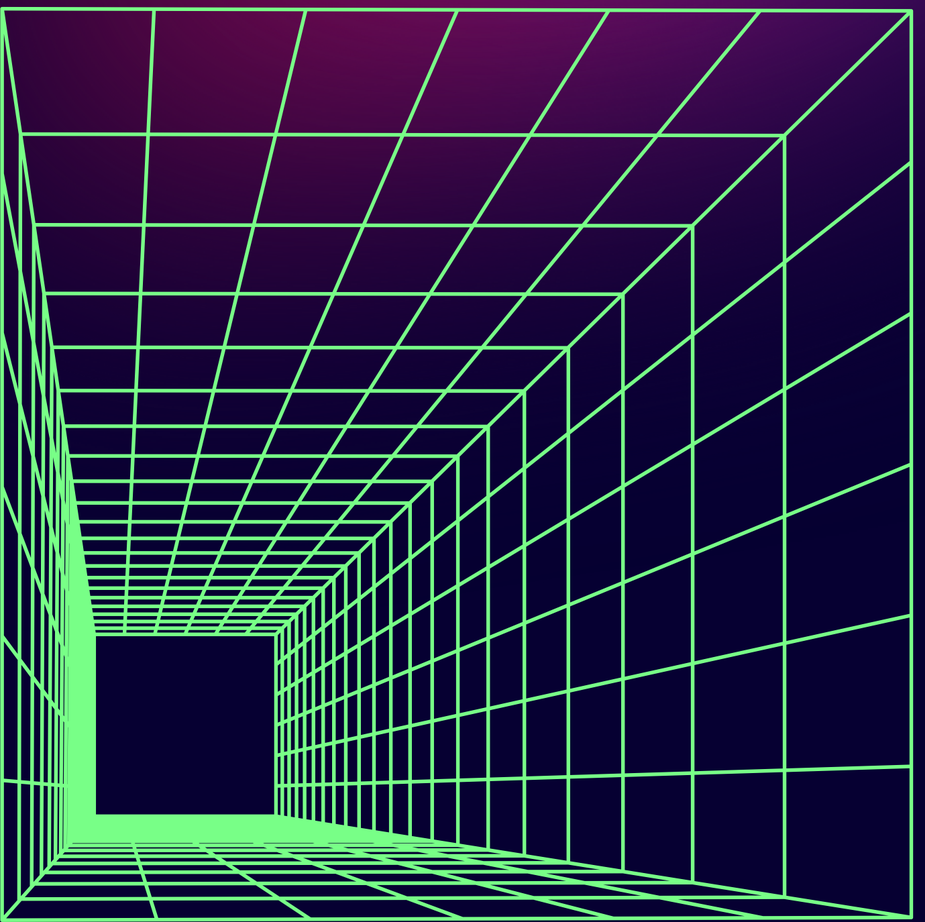
These are NP hard problems, difficult to find optimal solution.

Some Common approaches are :
1) Greedy Method
2) CELF Method

GREEDY

1

ALGORITHM



1. Defining Influence Spread: Establish a metric to quantify the extent of influence.
2. Initialization: Begin with a set of nodes that is initially empty.
3. Greedy Selection: During each iteration, calculate the incremental benefit of including each remaining node and choose the one with the greatest incremental gain.
4. Termination: Persist with the process until you reach a predetermined limit of nodes or until additional inclusions do not substantially enhance the influence spread.
5. Output: The chosen set of nodes serves as an approximate solution to the Influence Maximization Problem.

2 CELF

- Submodularity: Many outbreak detection objectives (detection time, population affected) are submodular - adding nodes improves gain efficiently.
    - **CELF Algorithm:** An efficient approximation algorithm exploiting submodularity to achieve near-optimal placements, Guarantees at least a constant fraction of the optimal solution, even with non-constant node costs.
    - Up to 700x faster than greedy algorithms.
    - **Real-world Applications:**
    - Water Quality Monitoring: Placing sensors in a water distribution network (millions of scenarios) to detect contamination.
    - Blogosphere Monitoring: Tracking information cascades in blog data (millions of posts) to prevent misinformation spread.

- The CELF (Cost-Effective Lazy Forward) algorithm is an improvement over the greedy algorithm for solving the Influence Maximization Problem (IMP). It was introduced to enhance the efficiency of the greedy algorithm by reducing the computational cost associated with evaluating the influence spread of candidate nodes.

- Lazy Evaluation:Unlike the greedy algorithm, CELF uses lazy evaluation to avoid recomputing the influence spread for all candidate nodes in each iteration.For each node, only recompute its marginal gain when necessary (i.e., when it becomes a potential top influencer).Maintain a set of “candidates” - nodes whose marginal gains need to be recomputed.

- Priority Queue:Nodes are added to the priority queue based on their current marginal gain estimates.The priority queue is initialized with all nodes, sorted by their marginal gains in descending order.

Depending on the time of detection $t = T(i, s)$, and the impact on the network before the detec tion (e.g., the size of the cascades missed, or the population affected by a contaminant), we incur penalty $\pi_i(t)$. The Research Track Paper 421 penalty function $\pi_i(t)$ depends on the scenario. We discuss concrete examples of penalty functions below. Our goal is to minimize the expected penalty over all possible scenarios i:

$$\pi(A) \equiv \quad i\ P(i)\pi_i(T(i, A))$$

Instead of minimizing the penalty $\pi(A)$, we can consider the scenario specific penalty reduction $R_i(A) = \pi_i(\infty) - \pi_i(T(i, A))$, and the expected penalty reduction

$$R(A) = \quad i\ P(i)R_i(A) = \pi(\varnothing) - \pi(A)$$

MAXIMIZING THE SPREAD OF INFLUENCE THROUGH A SOCIAL NETWORK1

Two basic diffusion models for the spread of influence: the Linear Threshold Model and the Independent Cascade Model.

- They are based on the idea that a node becomes active (adopts a new behavior or idea) if enough of its neighbors are active.

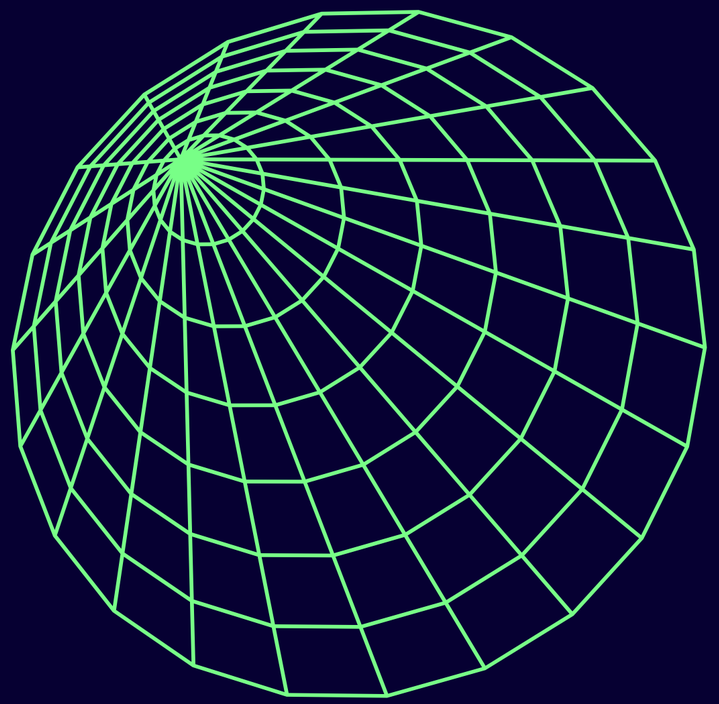
In the linear threshold model, each node has a random threshold that determines how many active neighbors it needs to become active.

In the independent cascade model, each edge has a random probability that determines how likely an active node can activate its neighbor. The probability is a fraction between 0 and 1, and it is independent of other edges.

Both models are used to study problems such as influence maximization, which aims to find the optimal seed set that can trigger the largest cascade of activations in the network.

Results: The influence maximization problem is NP-hard for both models, but can be approximated to within a factor of (1-1/e) by a greedy algorithm

FAIRNESS



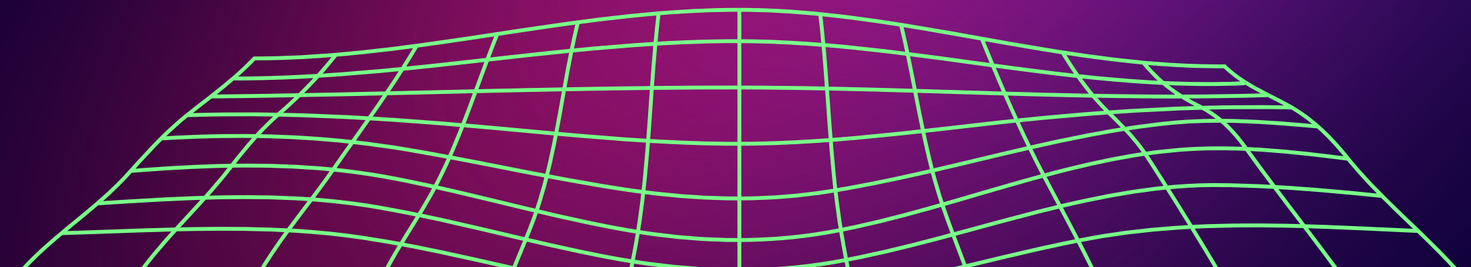
- Fairness refers to the quality of being just, equitable, and impartial in treatment or distribution. It entails treating individuals or groups without bias or discrimination.

- Human networks may encode historical biases, algorithms performing on them might capture and reproduce such biases when automating outcomes.

- Aim to understand the effect of communities in the creation of disparate impact among network participants based on demographic attributes.

FAIRNESS IN SOCIAL INFLUENCE MAXIMIZATION

- **Importance of Social Influence Maximization**: Algorithms for influencing social networks impact areas like advertising and news spread, making them crucial for understanding information flow.
- **Bias Reproduction Concern**: Existing algorithms may inadvertently perpetuate biases encoded within human networks, highlighting the need for fairness.
- **Demographic Blindness in Algorithms**: In theory, most early-adopter selection algorithms are blind to demographics, focusing solely on an individual's network position to assess their likelihood of early adoption. This approach assumes that network structure alone determines influence potential.



Fairness in seed selection

1. **Representation Across Demographics**: Fair seed selection ensures that the initial set of individuals chosen to propagate information or influence in a social network represents diverse demographic groups accurately.
2. **Bias Mitigation:** It aims to prevent the amplification of existing biases or inequalities within the network, ensuring that no particular group is unfairly favored or disadvantaged in the selection process.
3. **Efficiency and Effectiveness:** Fair seed selection balances the objective of fairness with the need to efficiently maximize influence or information spread, achieving both goals simultaneously.
4. **Continuous Improvement**: Achieving fairness in seed selection requires ongoing evaluation and refinement of algorithmic approaches to better align with fairness objectives and mitigate demographic disparities.

Group-Fairness in Influence Maximization

- **Diversity:** In our network, diversity refers to the presence of multiple groups, such as different ethnicities, genders, sexual orientations, or other relevant categories, within the population. Each individual in the network can identify with one or more of these groups, representing their diverse identities and affiliations.
- **Objective of Maximin Fairness:** Maximin Fairness aims to improve the conditions for the least well-off groups within a population. It focuses on maximizing the minimum influence received by any group, proportional to their population size.
- **Utility Function:** The utility function is defined as the minimum influence received by any group divided by the size of that group. The goal is to maximize this utility function subject to the constraint of maximin fairness.
- **Maximizing Overall Influence:** Under the maximin constraint, the objective is to maximize overall influence in the network.

- Group Rationality: Group Rationality extends the concept of individual rationality to ensure that no group is better off by leaving the influence maximization game and allocating resources internally. It aims to maintain fairness among groups by considering their proportional allocation of resources within the larger population.
- Diversity Constraints: The diversity constraints ensure that the influence received by each group in the network is at least equal to what they could achieve on their own with their fair allocation of seeds.This ensures that no group is disadvantaged in the influence maximization process.
- **Objective Function**: The objective function of the diversity constraint is to maximize the expected number of nodes activated, subject to the constraints imposed by ensuring group rationality.
- Price of fairness : Achieving fairness may sometimes come at the expense of efficiency or utility. For example, ensuring equal opportunities or outcomes for all individuals may require reallocating resources or making decisions that reduce overall efficiency.

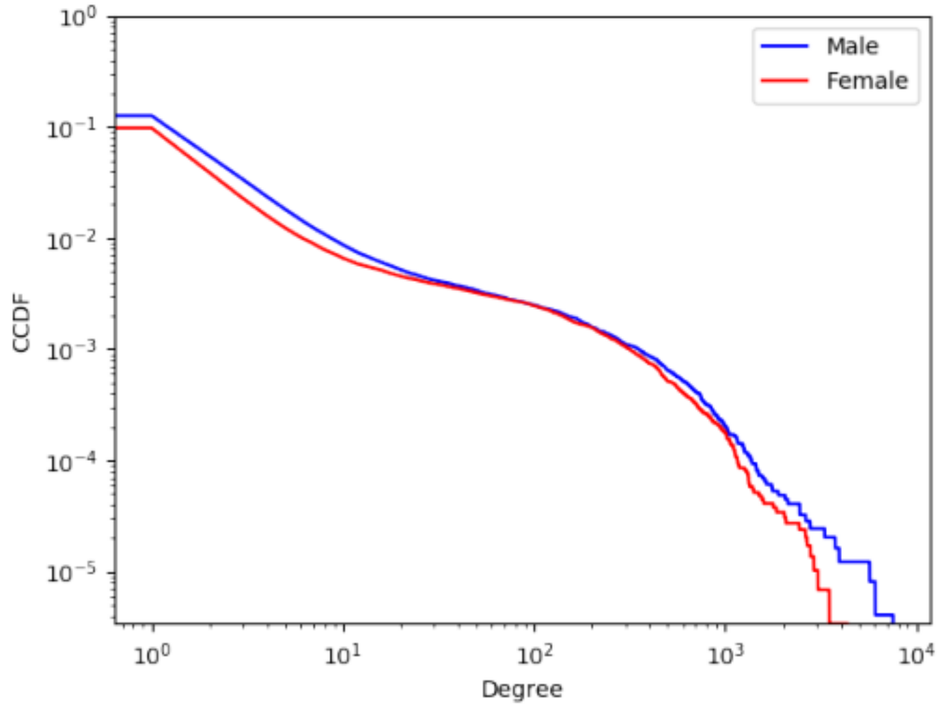


Figure 1: Complementary cumulative distribution function of the degrees in the Instagram graph, plotted for the males (blue) and the females (red).

Algorithm 1 Multiobjective Optimization$(\gamma, \tau, T, T', \eta)$

$S_1 = \{j : f_i(\{j\}) \geq \tau$ for some $i\}$
$x =$ MULTIFW$(k - |S_1|, \{\gamma\,(W_i - f_i(S))\}_{i=1}^m)$
$S_2 =$ SWAPROUND(x_{int}) //see [Chekuri *et al.*, 2010]
return $S_1 \cup S_2$

Algorithm 2 Multiobjective Frank-Wolfe$(k, \{W_i\})$

$x^0 = 0$
for $t = 1 \ldots T$ **do**
  $v^t =$ S-SP-MD$(x, \{i : W_i - F_i(x^{t-1}) \geq \epsilon\})$
  $x^t = x^{t-1} + \frac{1}{T} v^t$
return APPROXDECOMPOSITION(x^T) //see [Mirrokni *et al.*, 2017]
function S-SP-MD$(x, \mathcal{I})$
  Initialize v s.t. $||v||_1 = k$ and $y \in \Delta(\mathcal{I})$ arbitrarily
  for $\ell = 1 \ldots T'$ **do**
    Sample $i \sim y$; set $\hat{\nabla}_v = \frac{1}{W_i - F_i(x)} \mathcal{A}^i_{\text{grad}}(x)$
    Sample $j \sim v$; $\hat{\nabla}_y = k \cdot \text{diag}\left(\frac{1}{\vec{W} - \vec{F}(x)}\right) \mathcal{A}^j_{\text{item}}(x)$
    $y = \frac{y e^{-\eta \hat{\nabla}_y}}{||y e^{-\eta \hat{\nabla}_y}||_1}$
    $v = k \frac{\min\{v e_v^{\eta \hat{\nabla}}, 1\}}{||\min\{v e_v^{\eta \hat{\nabla}}, 1\}||_1}$

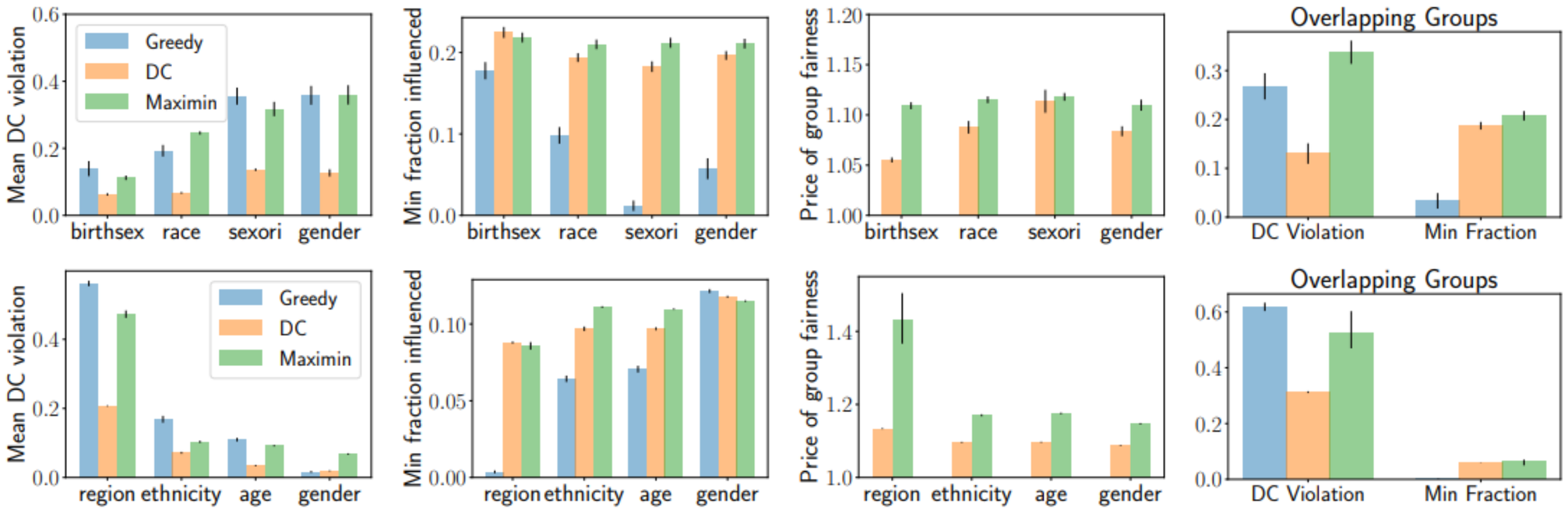


Figure 1: Average performance on homeless youth social networks (top) and simulated Antelope Valley networks (bottom).