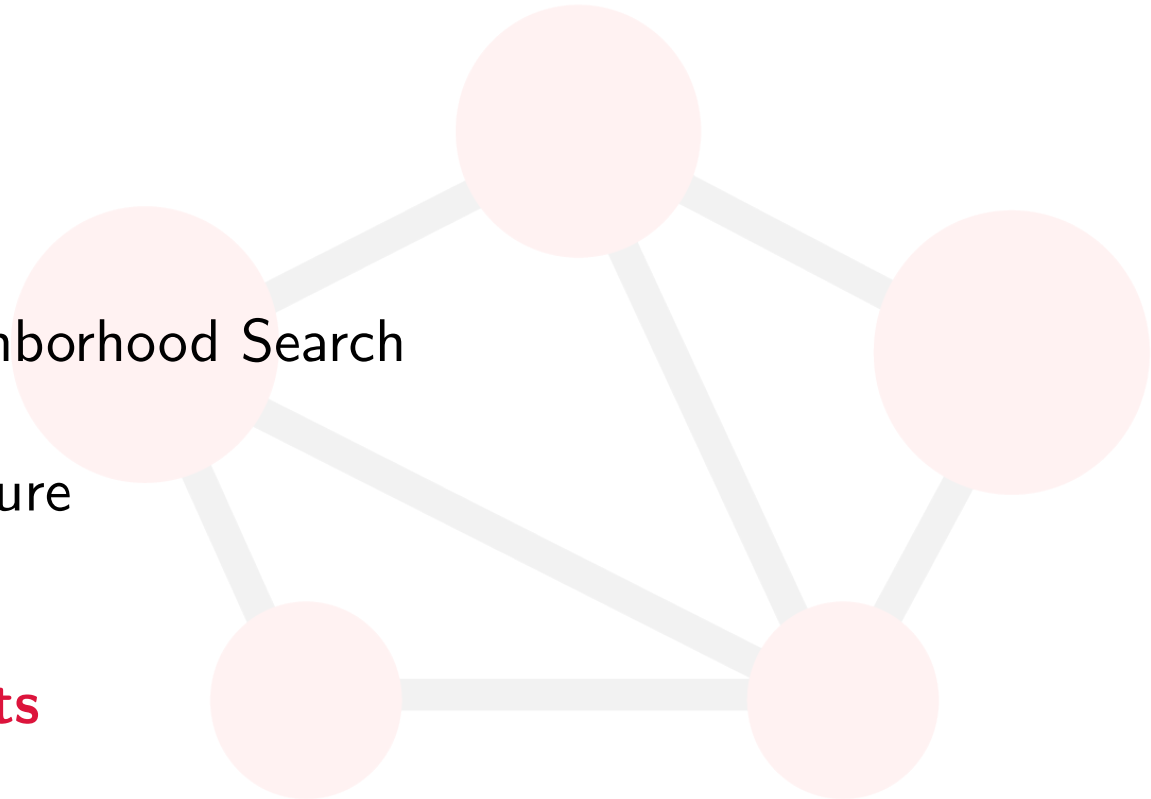
Solving the Minimum Positive Influence Dominating Set Problem in Social Networks Using Metaheuristics

XI International Conference Variable Neighborhood Search (ICVNS)

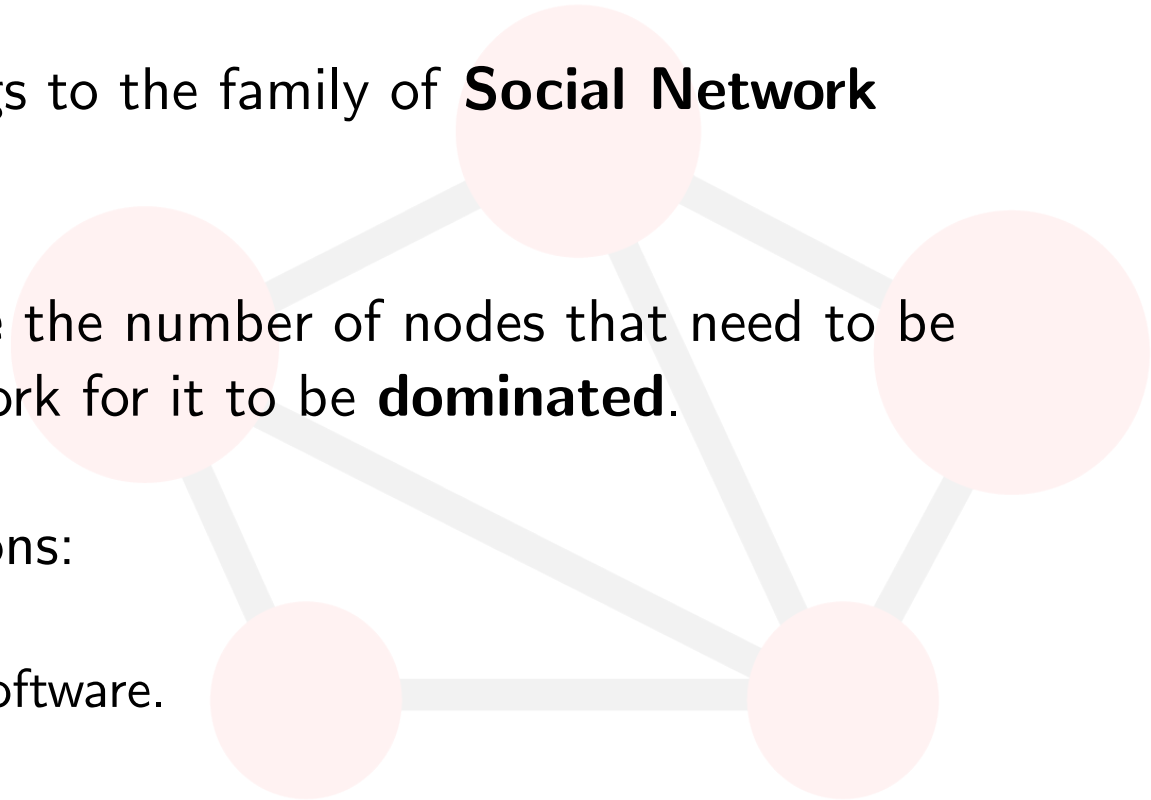
Iván Penedo (ivan.penedo@urjc.es)
Isaac Lozano-Osorio (isaac.lozano@urjc.es)
Jesús Sánchez-Oro (jesus.sanchezoro@urjc.es)
Óscar Cordón (ocordon@decsai.ugr.es)

Index



Introduction

- The **problem** belongs to the family of **Social Network Influence**.
- It seeks to **minimize** the number of nodes that need to be influenced in a network for it to be **dominated**.
- Real world applications:
  - Viral marketing.
  - Online learning software.

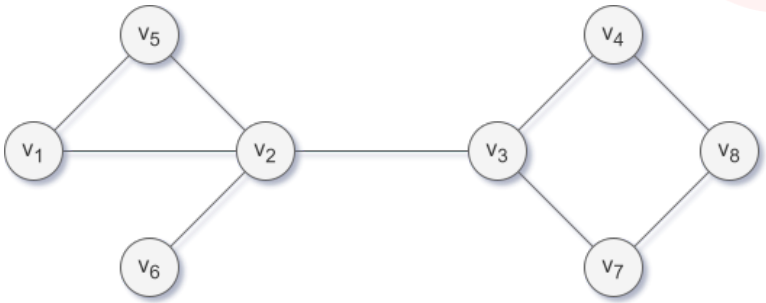


Introduction

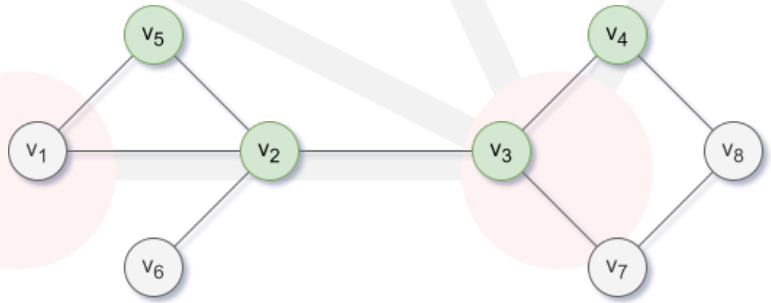
Formal problem definition

Given a social network represented by a graph $G = (V, E)$, the solution would be a **set of vertices** such that, by influencing them, the entire network becomes dominated.

To **dominate** a vertex v, **at least half of its neighboring vertices** $N(v)$ must be included in the solution set S.



(a) Graph with 7 nodes and 9 egdes.



(b) Solution S.

Introduction

Objective function

The goal consists of finding a subset of vertices S such that by influencing them, the entire network can be dominated, with the **minimum** cardinality.

$$\begin{aligned} \textbf{min.} \quad & \sum_{i=1}^{|V|} x_i \\ \textbf{s.t.} \quad & \sum_{v_j \in N(v_i)} x_j \geq \left\lceil \frac{|N(v_i)|}{2} \right\rceil \quad \forall i \in V \\ & x_i \in \{0,1\} \quad \forall i \in V \end{aligned} \tag{1}$$

Introduction

Literature review

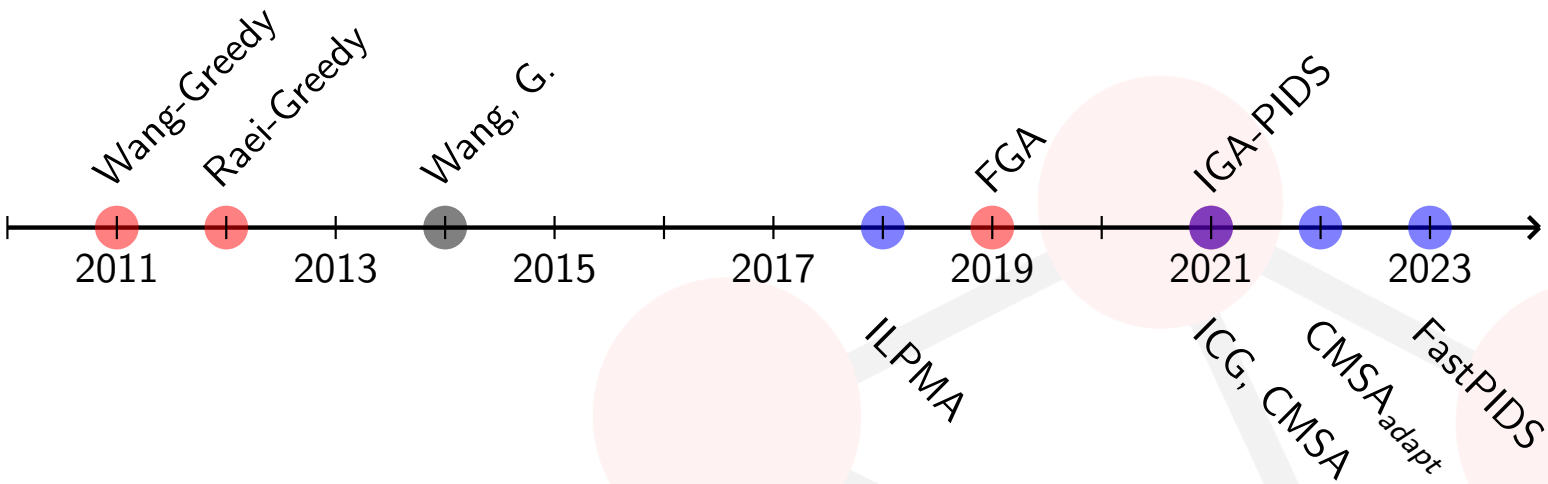


Figure 2: Timeline of the State of the Art.

Reviews of the state-of-the-art highlight studies on greedy algorithms and heuristics.

Minimum dominating set problems like the one studied are known to be $\mathcal{NP}$-**Hard**.

Proposal

Basic Variable Neighborhood Search (BVNS)

- Proposed by **Nenad Mladenovic** and **Pierre Hansen** in 1997.
- Main contributions:
  1. Consider several neighborhoods during the search.
  2. Perform systematic changes in the neighborhood structures.

Proposal

Basic Variable Neighborhood Search (BVNS)

Algorithm 1 *BVNS* $(G, \alpha, \delta, t_{\max}, k_{\max}) \rightarrow S$

$S_b \leftarrow V$
while $time \leq t_{\max}$ **do**
    $S \leftarrow Construct(G, \alpha)$
    $S \leftarrow LocalSearch(S, \delta)$
    $k \leftarrow 1$
    while $k \leq k_{\max}$ **do**
        $S' \leftarrow Shake(S, k, \alpha)$
        $S' \leftarrow LocalSearch(S', \delta)$
        $k \leftarrow NeighborhoodChange(S, S', k)$
    end while
end while
return S_b

Proposal

Shake Procedure in BVNS

In Basic Variable Neighborhood Search (BVNS), the perturbation mechanism is known as the **Shake** procedure.

We propose a Shake method that modifies the structure of the current solution based on a parameter k, whose value ranges from 1 to $k_{\max}$. This is a predefined input parameter of the overall procedure.

The proposed Shake performs:

- k **random removals** in the current solution set.
- A **reconstruction** of the solution using the existing construction method.

Proposal

Neighborhood reduction rules

Prior to the initial construction of a solution, an initialization process is carried out where some of the reductions proposed by **Sun et al.** are applied.

1. If a leaf node has a single parent node, the parent must always be included in the set S.
2. In a structure where nodes form a triangle through their relationships, the solution set S must contain two of the three nodes.

Proposal

Greedy construction

- Add vertices to the solution set using the **greedy criteria** based on degree.
- Process:
  - Select a vertex v based on its **degree** to dominate it.
  - Using the same criterion on the **neighbors of** v, add the required vertices to dominate the selected vertex.
  - The number of vertices needed to dominate vertex v is $max(0, \lceil \frac{deg(v)}{2} \rceil - |N \cap S|)$.
- At the end, a removal of redundant nodes is performed to reduce the cardinality of the solution while maintaining its feasibility.

Proposal

Randomized greedy construction

- The parameter α controls the balance between greediness and randomness during the construction phase.
- Some possible values for α:
  - $\alpha = 0$: **Fully Greedy**. The algorithm selects the best possible vertex at each step based on the degree.
  - $\alpha = 1$: **Fully Random**. The algorithm selects vertices randomly without considering their degree.
  - $\alpha = \mathit{RND}$: **Random α value**. The algorithm chooses randomly between fully greedy and fully random with probabilities controlled by α during execution. The value of α will fluctuate between 0 and 1 dynamically.
- The choice of α affects the balance between exploration and exploitation in the solution search.

Proposal

Piercing local search

Improve the solution through a **three-step** process for each vertex in the set S:

1. Generate a hole of size δ in a vertex $v \in S$. In this process, remove from the set S those unassigned nodes from the δ-neighborhood of the selected vertex.
2. Rebuild the solution using the previous constructive process with α being fully greedy.
3. Eliminate redundant nodes.

The best solution obtained during the process is recovered.

Proposal

Piercing local search

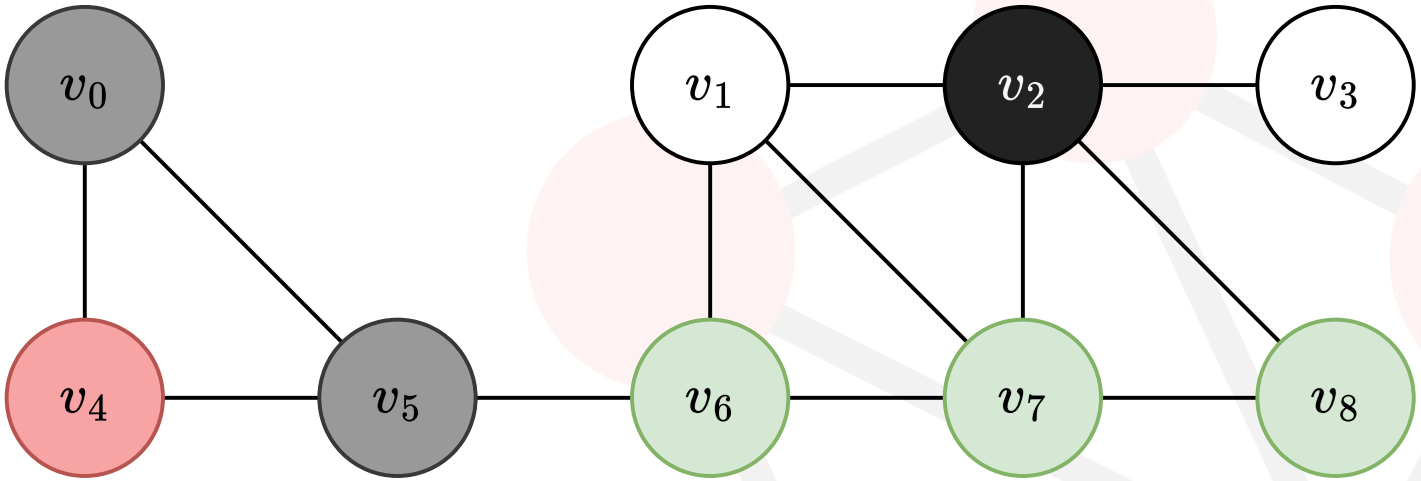


Figure 3: Example of a solution given by an initial construction.

Proposal

Piercing Local Search

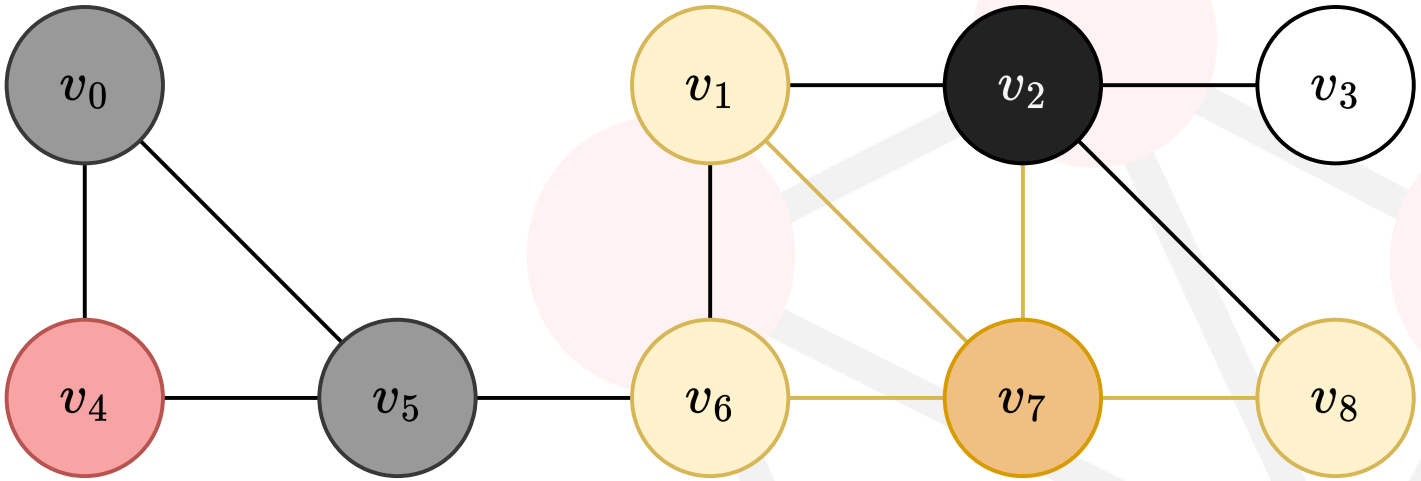


Figure 4: Propagation of Piercing Local Search with $\delta = 2$.

Proposal

Piercing local search

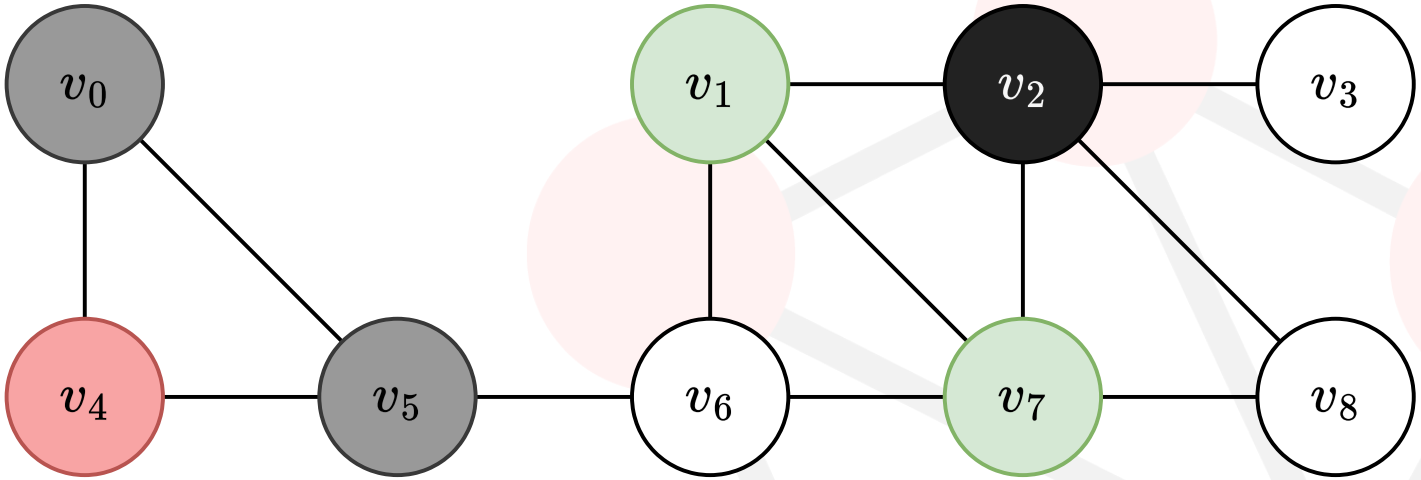


Figure 5: MPIDS Solution after Piercing Local Search.

Computational results

Environment

- Programming Language: **Java 21**.
- Experimental machine features: Ubuntu Server 20.04 with **128GB** RAM over **AMD EPYC 7282** using a single core.
- Performance metrics:
  - **Avg.**: the average objective function value.
  - **Time (s)**: execution time measured in seconds.
  - **Dev. (%)**: average deviation with respect to the best known solution in the experiment.
  - **# Bests**: the number of times that the algorithm is able to reach the best solution in the experiment.
- Instances:
  - Preliminary: from 34 to 59.216.211 **nodes**, and from 78 to 92.522.017 **edges**.
  - Final: from 34 to 59.216.211 **nodes**, and from 78 to 261.321.071 **edges**.

Computational Results

Local Search Configuration

δ	Avg.	Time (s)	Dev. (%)	# Bests
1	621659.84	**2.87**	4.25	3
2	565363.71	252.13	**0.13**	**40**
3	**565325.10**	675.54	0.54	13

Table 1: Comparison of Different Values for the δ Parameter in the Hole Generation Phase.

Computational Results

Randomized Greedy Construction Configuration

α	Avg.	Time (s)	Dev. (%)	# Bests
RND	593147.63	3600.23	0.44	5
0.00	**565363.71**	**252.13**	1.30	6
0.25	615990.47	3600.16	0.51	12
0.50	593060.12	3600.17	0.35	15
0.75	579615.39	3600.22	**0.25**	14
1.00	643084.12	3600.26	0.63	**24**

Table 2: Comparison of Different Configurations of the α Parameter in the Construction Phase.

Computational Results

BVNS Proposal vs. State of the Art

Algorithm	Avg.	Time (s)	Dev. (%)	# Bests
FastPIDS	**1023412.42**	**3600.00**	**0.01**	**189**
BVNS	1038446.79	3600.03	1.25	18

Table 3: Comparison of the BVNS Proposal vs. State of the Art.

Conclusions

Future work

- Obtain **optimal values** for some instances.
- Analyze other greedy criteria.
- Study the parameters using **IRACE**.
- Optimize the code.

Solving the Minimum Positive Influence Dominating Set Problem in Social Networks Using Metaheuristics

XI International Conference Variable Neighborhood Search (ICVNS)

Iván Penedo (ivan.penedo@urjc.es)
Isaac Lozano-Osorio (isaac.lozano@urjc.es)
Jesús Sánchez-Oro (jesus.sanchezoro@urjc.es)
Óscar Cordón (ocordon@decsai.ugr.es)