New metaheuristic algorithms for the analysis of the user influence in social networks

Isaac Lozano-Osorio, Jesús Sánchez-Oro, Óscar Cordón and Abraham Duarte.

isaac.lozano@urjc.es

jesus.sanchezoro@urjc.es

abraham.duarte@urjc.es

ocordon@decsai.ugr.es

Universidad Rey Juan Carlos

Outline

- Social Network Influence
- State of the art
- GRASP for SNIMP
- Results and Conclusions
- Future work

Social Network Influence

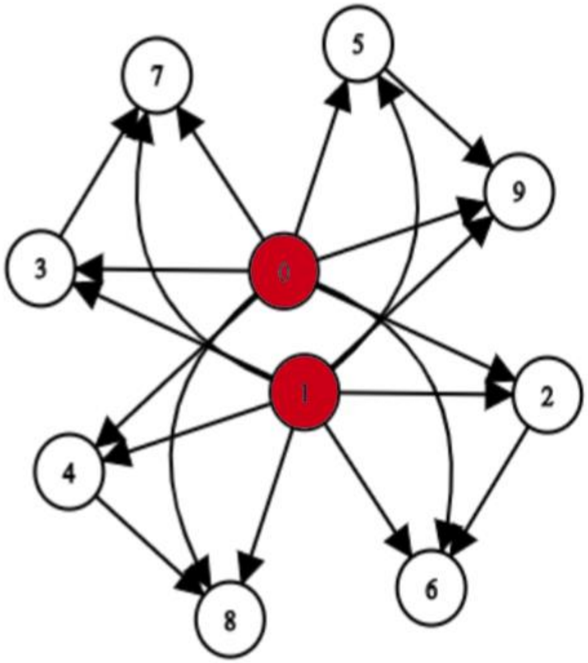
SOCIAL NETWORK INFLUENCE MAXIMIZATION PROBLEM

Finding the K most influential users in the social network with a simulation of an influence diffusion model.

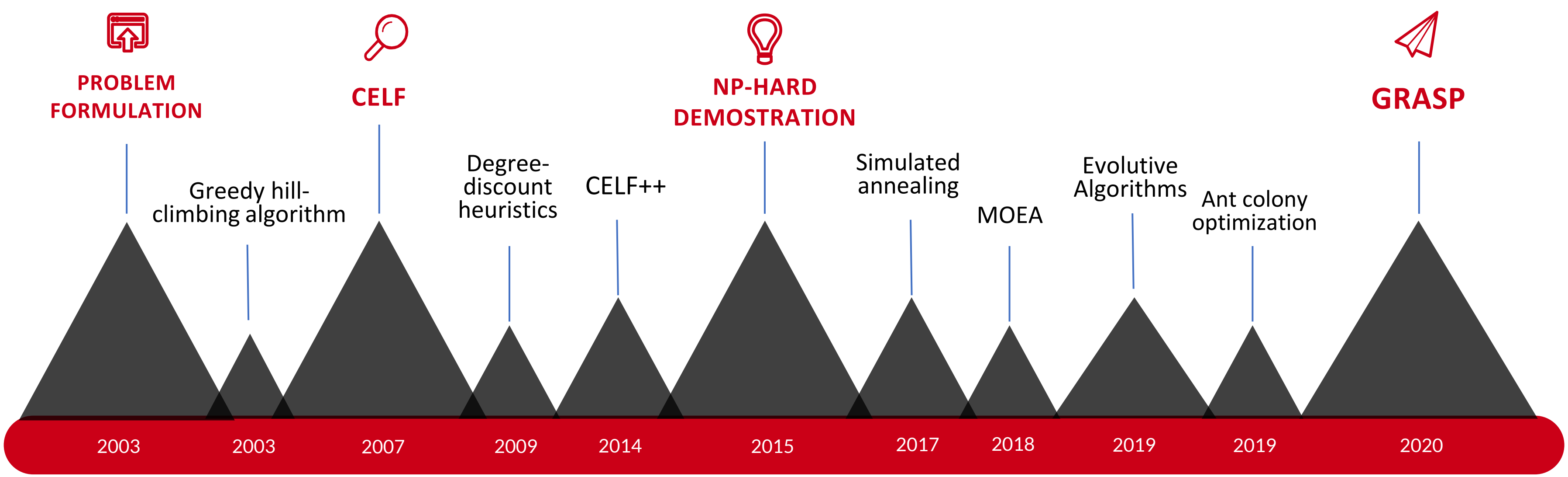
Commonly used for :

Viral marketing: Maximizing the number of impacts achieved during an advertising campaign.

Disease analysis: Finding the epicenter to eradicate the disease



State of the art



Influence Evaluation

- **INDEPENDENT CASCADE MODEL**
- **WEIGHTED CASCADE MODEL**
- **LINEAR THRESHOLD MODEL**

Influence Evaluation

MONTE CARLO ALGORITHM

Evaluation function, returns the number of influenced nodes.

Key steps of the algorithm :

Repetitions: The Monte Carlo algorithm is a probabilistic algorithm. N iterations are performed and the average result is kept.

Random values: Random values are generated in step 9 for each influenced element. If value is higher or equal to ***p*** the element will be influenced.

Algorithm 1 $ICM(G = (V, E), S, P, IT)$

```
1: spread ← ∅
2: for i ∈ 1 ... IT do
3:     A* ← S
4:     A ← S
5:     while A ≠ ∅ do
6:         B ← ∅
7:         for v ∈ A do
8:             for (u, v) ∈ E do
9:                 if rnd(0, 1) ≥ p then
10:                    B ← B ∪ u
11:                end if
12:            end for
13:        end for
14:        A* ← A* ∪ B
15:        A ← B
16:    end while
17:    spread ← spread + |A*|
18: end for
19: return spread/IT
```

Objectives of the work

STATE OF THE ART PROBLEMS

- ☒ Multiple Monte Carlo evaluations.
- ☒ Slowness.

KEY POINTS OF THE WORK

- ☐ Reduction of Monte Carlo evaluations with greedy algorithms.
- ☐ Surrogate local search limiting searches.
- ☐ Comparison with different algorithms of the state of the art.

Greedy Randomize Adaptative Search Procedure

Constructive Phase

CONSTRUCTIVE PHASE

It generates an initial solution and is generally guided by a greedy selected function.

Heuristic functions used:

Closeness coefficient: Measures the average distance to all other nodes.

Clustering coefficient: The measure of the degree to which nodes in a graph tend to cluster.

Greedy heuristic algorithm: Based on the first and second degree of neighbors of a user.

Algorithm 2 $GRASP(G = (V, E))$

$S \leftarrow \emptyset$
$CL \leftarrow \text{V}$
while $|S| < K$ **do**
    $g_{min} \leftarrow min_{u \in CL} g(u)$
    $g_{max} \leftarrow max_{u \in CL} g(u)$
    $\mu \leftarrow g_{max} - \alpha \cdot (g_{max} - g_{min})$
    $RCL \leftarrow v \in CL : g(v) \geq \mu$
    $u \leftarrow rnd(RCL)$
    $S \leftarrow S \cup \{u\}$
    $CL \leftarrow CL \setminus \{u\}$
end while
return S

Local Search Phase

LOCAL SEARCH PHASE

Improves the solution generated by the construction phase in order to reach a local optima.

Local search used:

Exchanges with N number of neighbors to explore (MNV): Selection of the exact number of neighbors where the exchange is performed.

Exchanges with N percentage of nodes to explore (PCV): Exploration of a given percentage of neighbors.

Algorithm 3 $SURROGATE - LS(S, C)$

```
firstImprovement ← True
bestMark ← evaluate(S)
while firstImprovement do
    firstImprovement ← False
    for node in S and !firstImprovement do
        for candidate in C and !firstImprovement do
            Saux ← swap(S,node,candidate)
            if evaluate(Saux) > bestMark then
                firstImprovement ← True
                bestMark ← evaluate(Saux)
                S ← Saux
            end if
        end for
    end for
end while
return S
```

Results and Conclusions

Instances

	Nodes	Edges	Diameter
p2p-Gnutella31	**62586**	**147892**	**11**
ca-AstroPh	**18772**	**198110**	**14**
ca-CondMat	**23133**	**93497**	**14**
cit-HepPh	**34546**	**421578**	**12**
email-Enron	**36692**	**183831**	**11**
email-EuAll	**265214**	**420045**	**14**

We have used 6 different networks from SNAP
https://snap.stanford.edu/data/

Heuristic selection experimentation

	OF	Time(s)	Dev(%)	Best
ALG - HEUR	**490,80**	**14,69**	**0,00**	**10/10**
ALG - CLUS	**356,10**	**81,14**	**34,33**	**0/10**
ALG – CLOSS*	**459,57**	**5,68**	**43,88**	**0/10**

* The algorithm did not finish due to memory limits.

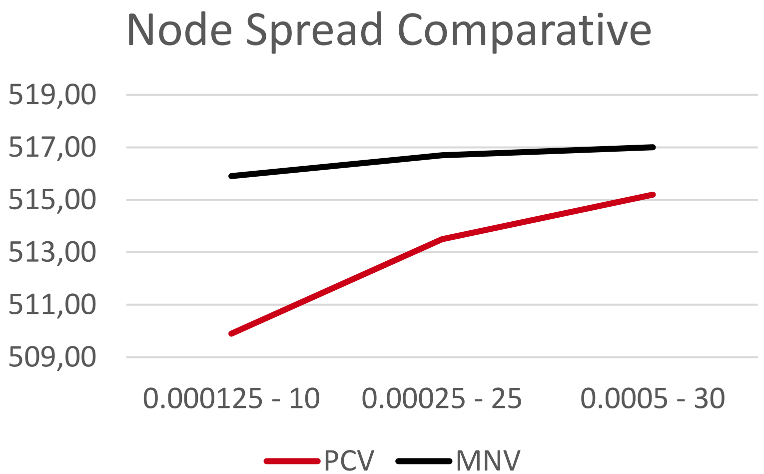
OF: the average objective function value

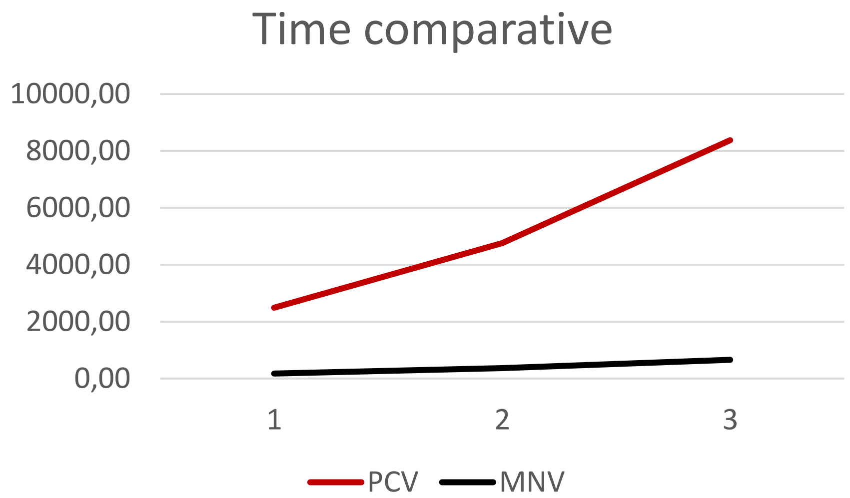
Time (s): the average computing time required by the algorithm in seconds

Dev(%): the average deviation with respect to the best value found in the experiment

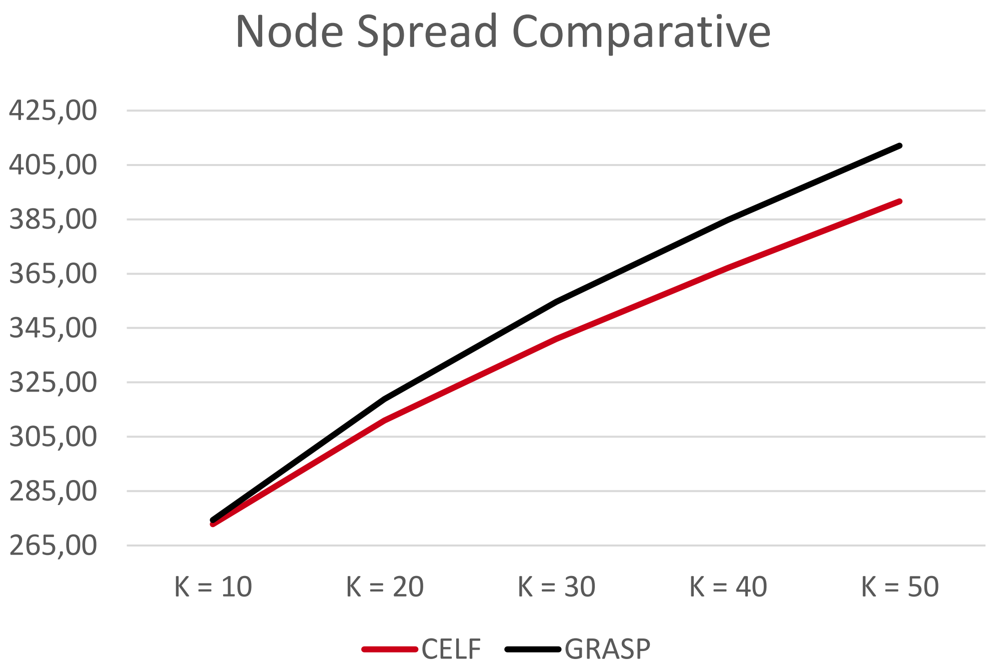
Best: the number of times that the algorithm matches the best solution

Local search selection experiments

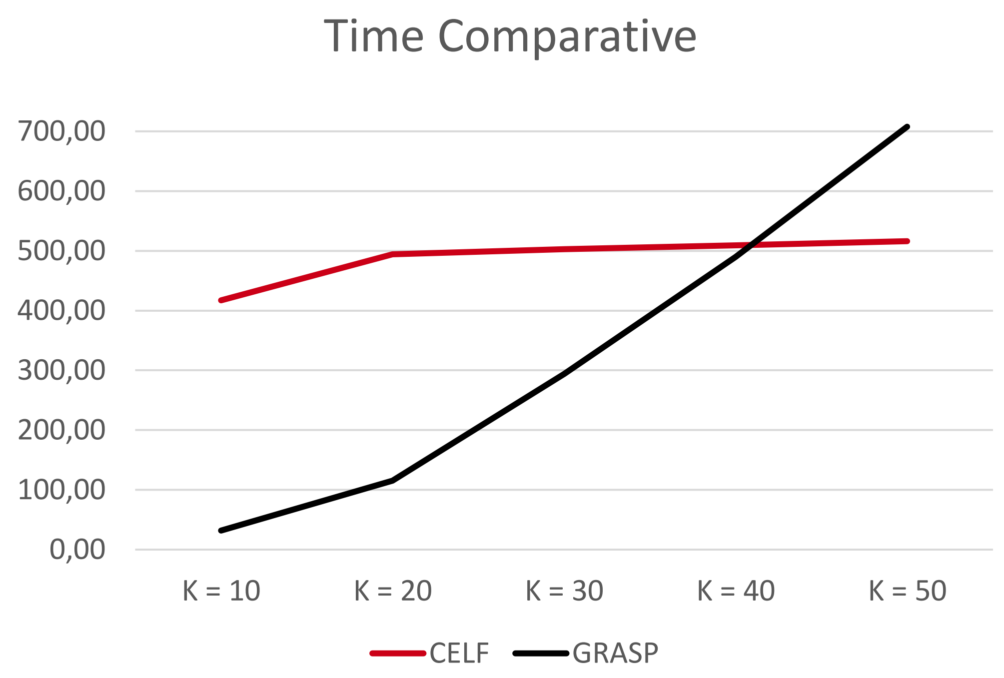




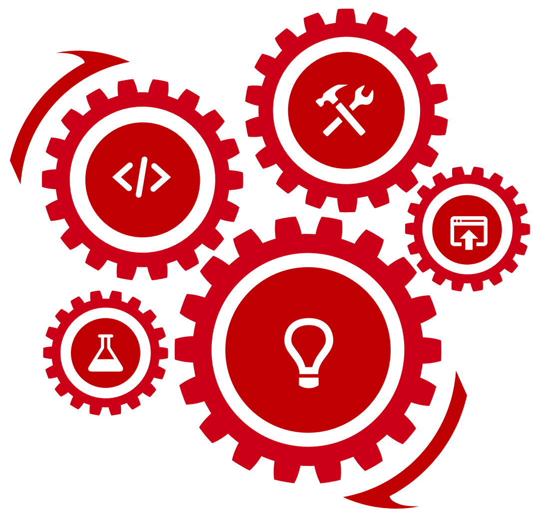
Results


Node Spread Comparative
425,00
405,00
385,00
365,00
345,00
325,00
305,00
285,00
265,00
K = 10
K = 20
K = 30
K = 40
K = 50
CELF
GRASP

Results


Time Comparative
700,00
600,00
500,00
400,00
300,00
200,00
100,00
0,00
K = 10
K = 20
K = 30
K = 40
K = 50
CELF
GRASP

Future Work



NEW ALGORITHMS

Improve performance both in quality and time.

TIME OPTIMIZATION

Depth in algorithm to reduce overall time.

MORE INSTANCES

Use more instances to compare our algorithm with state of the art.

STATE OF THE ART

Compare the performance of the algorithm against more algorithms from SoA.

New metaheuristic algorithms for the analysis of the user influence in social networks

Isaac Lozano-Osorio, Jesús Sánchez-Oro, Óscar Cordón and Abraham Duarte.

isaac.lozano@urjc.es

jesus.sanchezoro@urjc.es

abraham.duarte@urjc.es

ocordon@decsai.ugr.es

Universidad Rey Juan Carlos