Research Report

GT-NDNetw: Game Theoretical Approaches for the Critical Node Detection Problem in Social and Economic Networks

PN-III-P1-1.1-TE-2019-1633
56 / 2020

Phase 1

Objective 1. Development of new algorithms for the CND problem based on game theory

A 1.1. Documentation of existing algorithms in the literature; (I.1)

A 1.2. Documentation of existing connectivity metrics σ; (I.1, I.2, I.3, I.4)

A 1.3. Documentation of viral networks, financial networks; (I.3)

A 1.4. Development of new algorithms based on non-cooperative game equilibria and on cooperative games;(I.3, I.4)

A 1.5. Comparison with existing state-of-the-art methods on benchmark networks; (I.4)

A 1.6. Application on viral and economic networks, and analysis of the obtained critical nodes; (I.3)

Deliverables:

- research report;
- articles submitted for publication: Eliézer Béczi, Noémi Gaskó, A memetic algorithm for the bi-objective critical node detection problem, submitted to Applied Intelligence
- web page: www.cs.ubbcluj.ro/˜gaskonomi/GT-NDNetw

I Implementation

The main contributions of the project are presented in what follows.

I.1 Critical node detection problem

In recent years, complex networks received a lot of attention due to their applicability in various domains. Several optimization problems were studied within complex networks like community detection [9], maximal influence node detection [11], link prediction [17] etc. All the aforementioned problems reveal major insights in the studied networks.

Identifying critical nodes (critical node detection problem - CNDP) in a complex network is also a crucial task, the base problem consisting in minimizing pairwise connectivity by removing a subset of K nodes. In [3] it was proven that it is an NP-hard problem.

The general formulation of the problem is the following [14]: given a $G = (V, E)$ graph and a connectivity metric σ, find the set of nodes $S \subseteq V$ such that $G[V\backslash S]$ satisfies the metric σ. This metric is usually defined as an objective function which needs to be optimized (for example, maximizing the number of components, minimizing the component size, etc.).

CNDP has a large field of applicability, for example, in social network analysis [8], in epidemic control [21], network immunization [12], and biological networks [18].

Several algorithms were designed for the CNDP, with most exact methods based on the integer linear programming formulation of the problem [7]. In [1] a dynamic programming approach is proposed for a special class of graphs. As approximation algorithms we can mention for example, a simulated annealing algorithm [22]. A thorough survey about existing methods for the CNDP can be found in [14].

The CNDP has several variants that explore the connectivity metric λ. Other variants with constraints were introduced, such as the cardinality constrained critical node detection (CC-CNDP) [4], component-cardinality-constrained

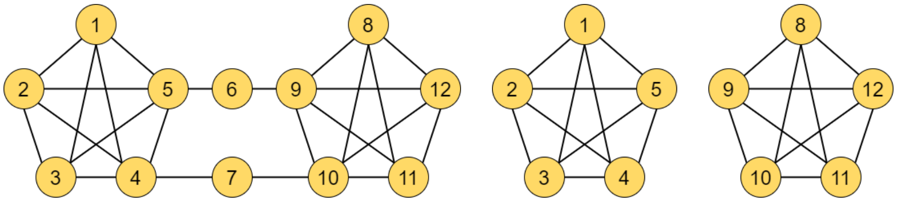


Figure 1: Example of a simple graph (left figure), if we delete nodes 6 and 7 the graph will be fragmented in two components (right figure) (source of the image [2]).

Algorithm 1 iGreedy

```
Require: $G, k$
$S \leftarrow$ VERTEX COVER$(G)$
while $|S| > k$ do
    $B \leftarrow \arg\min_{i \in S} f(S \setminus \{i\})$
    $S \leftarrow S \setminus \{$SELECT$(B)\}$
end while
return $S$
```

critical node problem (3C-CNDP) [13]. Also bi-objective variants of the CNDP exist, one of them is proposed in [16], in this variant the cost of removing the node counts. Another bi-objective variant is proposed in [23], which is in detail described in Section I.4.

I.2 σ - Pairwise connectivity

If H represents the set of maximal connected components, we can introduce:

$$f(S) = \sum_{h \in H} \frac{|h| \cdot (\|h| - 1)}{2}, \tag{1}$$

as the **pairwise connectivity** [22, 2]. The problem consists in minimizing the function 1:

$$\min_{S \subseteq V} f(S). \tag{2}$$

Example In the case of Figure 1, if we need to identify $k = 2$ critical nodes, then $S = \{6, 7\}$ gives the optimal solution. A $G[V \setminus S]$ graph will be fragmented in two connected components, $|H| = 2$. The function 1 will be calculated as follows:

$$f(S) = \frac{5 \cdot (5 - 1)}{2} + \frac{5 \cdot (5 - 1)}{2} = 20.$$

I.2.1 Algorithms

Next, we present three algorithms to solve the CNP problem. The first method is a simple greedy heuristic, the second is a genetic algorithm, and the third algorithm is a memetic approach, using the greedy algorithm in the initialization step of the genetic algorithm.

Greedy algorithm A simple greedy algorithm is described in Algorithm 1.

Genetic algorithm For the genetic algorithm, we modify the fitness function in order to get feedback from the existing solutions.

Fitness function The fitness function is:

$$g(S, S^*) = f(S) + \gamma \cdot |S \cap S^*|, \tag{3}$$

where S^* represents the best found solution, γ is a variable which has the role to maintain the diversity in the population. γ is calculated as follows:

$$\gamma = \frac{\alpha \cdot f(S^*)}{\langle |S \cap S^*| \rangle_{S \in P}}, \tag{4}$$

Algorithm 2 Genetic Algorithm

```
Require: $G, k, N, \pi_{\min}, \pi_{\max}, \Delta\pi, \alpha, t_{\max}$
$t \leftarrow 0$
Init($N, P, S^*, \gamma, \pi$)
while $t < t_{t_{\max}}$ do
    $P' \leftarrow$ Crossover($k, N, P$)
    $P' \leftarrow$ Mutation($k, N, P', \pi$)
    $P \leftarrow$ Selection($N, P, P'$)
    $S^*, \gamma, \pi =$ Update($N, P, S^*, \pi, \pi_{\min}, \pi_{\max}, \Delta\pi, \alpha$)
    $t \leftarrow t + 1$
end while
return $P$
```

α is a parameter describing the importance of the variables. Calculation of S^*, γ and π are outlined in Algorithm 3.

Representation For the genetic algorithm we used integer representation.

The outline of the genetic algorithm is presented in Algorithm 2. The main steps are outline din Algorithm 4 - the initialization, 6 - the mutation, Algorithm 5 - the crossover, and Algorithm 7 -the selection.

Algorithm 3 Update S^*, γ and π variables

```
Require: $N, P, S^*, \pi, \pi_{\min}, \pi_{\max}, \Delta\pi, \alpha$
$avg \leftarrow 0$
for $i \leftarrow 1, N$ do
    $S \leftarrow P[i]$
    $avg \leftarrow avg + |S \cap S^*|$
end for
$avg \leftarrow \frac{avg}{N}$
$\gamma \leftarrow \frac{\alpha \cdot f(S^*)}{avg}$
$S \leftarrow P[0]$
if $f(S) < f(S^*)$ then
    $S^* \leftarrow S$
    $\pi \leftarrow \pi_{\min}$
else
    $\pi \leftarrow \min(\pi + \Delta\pi, \pi_{\max})$
end if
return $S^*, \gamma, \pi$
```

Algorithm 4 Initialization

```
Require: $G, k$
$S \leftarrow V$
while $|S| > k$ do
    $elem \leftarrow$ Select($S$)
    $S \leftarrow S \setminus \{elem\}$
end while
return $S$
```

Memetic algorithm The memetic algorithm (MA) uses a smart initialization as the first step of the genetic algorithm, as described in Algorithm 8.

Algorithm 5 Recombination Operator

```
Require: $k, N, P$
$P' \leftarrow \emptyset$
for $i \leftarrow 1, N$ do
    $S_1 \leftarrow$ SELECT($P$)
    $S_2 \leftarrow$ SELECT($P$)
    $S' \leftarrow S_1 \cup S_2$
    if $|S'| = k$ then
        $P' \leftarrow P' \cup \{S'\}$
    else
        $S' \leftarrow$ RANDOM SAMPLE($S', k$)        ▷ Take $k$ random elements from $S'$
        $P' \leftarrow P' \cup \{S'\}$
    end if
end for
return $P'$
```

I.2.2 Numerical experiments

Parameter setting for the iGA: $N = 20$ is the size of the population, $\pi_{\min} = 50$, $\pi_{\max} = 90$ és $\Delta\pi = 2.5$, $\alpha = 2$, $t_{\max} = 100$ the maximum iteration number.

As benchmarks we used synthetic graphs described in Table 3. As shown in Table 1 in almost all instances the memetic algorithm outperforms the other two algorithms.

I.3 σ - Number of connected components

For this, the optimization problem is to maximise the number of connected components of $G[V - S]$, where S represents the set of the deleted nodes.

A non-cooperative game theoretical approach Our method consists in a game theoretical approach.

We consider a game in normal form $\Gamma = (N, S, U)$, where:

- $N = \{1, ..., n\}$ is the set of players,
- s_i is the strategy of player i, $S = S_1 \times ... \times S_n$ is the set of strategy profiles of the game
- $u_i : S \to \mathbb{R}$ is the payoff of player i, and $U = (u_1, \ldots, u_n)$ are the players' utility functions

In our case, the set of players are nodes, the strategy consists in being or not a critical node and the payoff function is the number of connected components in which the graph will be fragmented without the certain node.

Generative relations characterize a certain equilibrium type by using the non-dominance concept. A binary relation τ is defined on S, $\tau = (S, S, G \subset S \times S)$. If we have $s\tau q$ (i.e. $(s, q) \in G$) then we say that s dominates q with respect to relation τ. If for s^*, $\nexists q$ such that $q\tau s^*$, we call s^* non-dominated with respect to relation τ. The set containing all $s^* \in S$ that are non-dominated with respect to relation τ is called the non-dominated set of S with respect to τ.

Relation τ is called *generative* for an equilibrium type of a game (e.g. Nash, strong Nash, etc.) if the set of non-dominated strategy profiles with respect to relation τ equals that set of equilibria of the game.

Table 1: Results for the CNDP problem for test files.

	Greedy		GA		MA	
Graph	μ	σ	μ	σ	μ	σ
BA500	**195.90**	**0.88**	4,527.50	682.97	196.20	1.23
ER500	2,830.60	446.61	55,113.00	1,481.57	**2,278.40**	**138.52**
FF250	219.50	9.83	7,972.00	740.66	**215.40**	**5.80**
WS250	10,610.50	3,797.63	16,021.10	172.24	**8,039.90**	**838.95**

Algorithm 6 Mutation Operator

```
Require: $k, N, P, \pi$
$P' \leftarrow \emptyset$
for $i \leftarrow 1, N$ do
    $r \leftarrow$ RAND INT$(1, 100)$
    if $r \leq \pi$ then
        $S' \leftarrow P[i]$
        $n_g \leftarrow$ RAND INT$(0, k)$                ▷ Number of genes to mutate
        for $j \leftarrow 1, n_g$ do
            $elem \leftarrow$ SELECT$(S')$
            $S' \leftarrow S' \setminus \{elem\}$
        end for
        $MIS \leftarrow V \setminus S'$
        while $|S'| < k$ do
            $elem \leftarrow$ SELECT$(MIS)$
            $S' \leftarrow S' \cup \{elem\}$
        end while
        $P' \leftarrow P' \cup \{S'\}$
    else
        $S \leftarrow P[i]$
        $P' \leftarrow P' \cup \{S\}$
    end if
end for
return $P'$
```

Algorithm 7 Selection Operator

```
Require: $N, P, P'$
$P \leftarrow P \cup P'$
SORT$(P)$                ▷ Sort individuals by fitness function in ASC order
return $P[:N]$                ▷ Take best $N$ solutions
```

An easy way to construct a generative relation is to use a quality indicator $t : S \times S \rightarrow N$ such that: $s\tau q \Leftrightarrow t(s, q) < t(q, s)$

A generative relation for the Nash equilibrium was introduced in [19]:

Consider two strategy profiles s and q in S. Operator $\tau : S \times S \rightarrow \mathbb{N}$ defined as

$$t(s, q) = card\{i \in N | u_i(s) < u_i(q_i, s_{-i}), q_i \neq s_i\}$$

characterizes the NEs of the game.

The relation between the two strategy profiles s and q is:

- either s dominates q $(t(s, q) < t(q, s))$
- either q dominates s $(t(q, s) < t(s, q))$
- s and q are indifferent $(t(q, s) = t(s, q))$

Extremal optimization Extremal optimization (EO) [5] is a powerful optimization tool for combinatorial optimization problems for which a solution s can be expressed as a set of components $s_i, i = 1, \ldots, n$ with individual fitnesses $f_i(s)$ assigned to each one of them. The standard EO maximizes each component of a potential solution by randomly reassigning the one having the worst fitness. An individual s_{best} is used to preserve the best solution found so far based on an overall fitness $f()$. An outline of the standard EO is presented in Algorithm 9.

The used algorithm for the Nash equilibrium detection is the Nash Extremal Optimization [20], the main steps are outlined in Algorithm 10.

Algorithm 8 Smart Initialization

```
Require: $G, k, N$
$P \leftarrow \emptyset$
for $i \leftarrow 1, N \cdot \frac{10}{100}$ do
    $P \leftarrow P \cup \{\textsc{Greedy}(G, k)\}$
end for
while $|P| < N$ do
    $P \leftarrow P \cup \{\textsc{Rand Sol}(k)\}$
end while
return $P$
```

Algorithm 9 Standard EO

```
Initialize $s$ random;
$s_{best} := s$; //$s_{best}$ preserves the best solution found so far
while a termination condition is not met do
    find $i_{min}$ component with the smallest fitness value
    randomly reassign $s_{i_{min}}$ in $s$;
    if $f(s) > f(s_{best})$ then
        $s_{best} := s$
    end if
end while
```

Experimental results Experimental results illustrate the behaviour of EO and NEO by using the following set of networks: yeast[1], BA500, BA1000, ER250. For the numerical experiments, we used the EO and NEO, with random initialization and with a greedy initialization. Numerical results are presented in Table 2. Results indicate the potential of the proposed methods.

Case study - a financial stock market network To test the algorithms we used a financial stock market network [15], which contains the network obtained from the analysis of temporal correlations among the time-series of 62 stocks in the New York Exchange Market from the period 2012-2014. We use an unweighted version of the graph, we deleted all edges, where the weight was under the value 1.2.

As results we obtained the most ten critical nodes the following stocks: AEP - American Electric Power, ARC - ARC Document Solutions, AVP - Avcorp Industries Inc., ETR - Entergy , MRK - Merck & Co., PG - Procter & Gamble, SO - Southern Company, S, UIS - Unisys, WMT - Walmart. Figure 2 presents the network and the critical nodes.

I.4 Bi-objective critical node detection problem

Let $G = (V, E)$ be an undirected graph, where V is the set of nodes, and E is the set of edges.

The bi-objective critical node detection problem (BOCNDP) was proposed in [23] and consists in finding k nodes, that if we delete from graph G, we would like to optimize these two objectives:

[1] http://archive.ics.uci.edu/ml/datasets/yeast, last accessed 10.10.2020

Algorithm 10 Nash Extremal Optimization

```
Randomly intialize $s = (s_1, s_2, \ldots, s_n)$, $s_{best} = s$;
while termination criteria is not fulfilled do
    For the 'current' configuration $s$ evaluate $u_i$ for each player $i$;
    find $j$ satisfying $u_j \leq u_i$ for all $i, i \neq j$, i.e., $j$ has the 'worst payoff';
    change $s_j$ randomly;
    if $s$ dominates in Nash sense $s_{best}$ then
        $s_{best} = s$
    end if
end while
```

Table 2: Numerical results of the CNDP with four algorithms: EO, NEO, greedy initialization of EO based on degree distribution, greedy initialization of NEO

Graph	k	EO	NEO	Greedy+EO	Greedy+NEO
yeast	202	843	838	847	846
BA500	50	313	311	313	312
BA1000	75	590	588	590	589
ER250	50	68	65	68	65

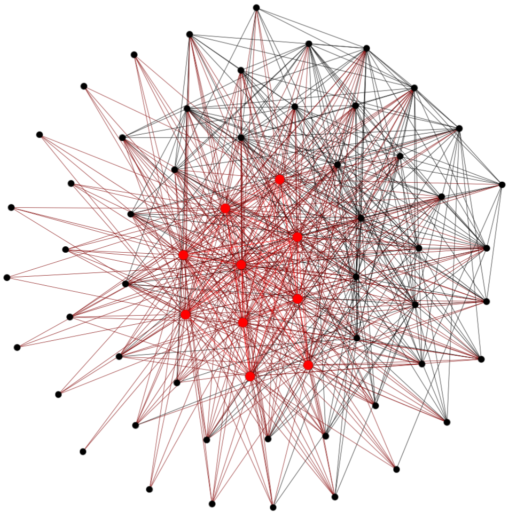


Figure 2: Stocks network, edges with weight ≥ 1.2 are kept, nodes colored in red are the most influential.

1. maximize the number of connected components
2. minimize the variance of the cardinality of the connected components

Formally, the objectives are the following:

$$\max \quad |H|, \tag{5}$$

$$\min \quad var(H), \tag{6}$$

$$\text{such that} \sum_{i \in S} w_i \leq W, \tag{7}$$

where w_i are the weights associated to the vertices of the graph and $W > 0$ is a constraint, H denotes $G[V \setminus S]$ the set of the connected components and $var(H)$ denotes the variance of the cardinality of the connected components and can be calculated with the following formula:

$$\frac{1}{|H|} \sum_{h \in H} \left(|h| - \frac{n^*}{|H|} \right)^2, \tag{8}$$

where $n^* = \sum_{h \in H} |h|$ is the number of nodes in $G[V \setminus S]$. The BOCNDP is distinct from the CNDP [23].

Example Let us consider a simple example, the graph presented in Figure 3. If we need to identify $k = 2$ critical nodes, then $S = \{2, 3\}$ is the optimal solution. The $G[V \setminus S]$ will have 5 components, $|H| = 5$ and

$$var(H) = \frac{1}{5} \cdot \left[\left(1 - \frac{13}{5}\right)^2 + 4 \cdot \left(3 - \frac{13}{5}\right)^2 \right] = \frac{16}{25} = 0.64.$$

Because the BOCNDP is a relatively new problem formulation, the literature is not too reach in the proposed algorithms. In [23] six existing multi-objective algorithms are used to resolve the BOCNDP. In [16] a different variant of the BOCNDP, called Bi-CNDP is introduced and studied with decomposition-based multi-objective evolutionary algorithms.

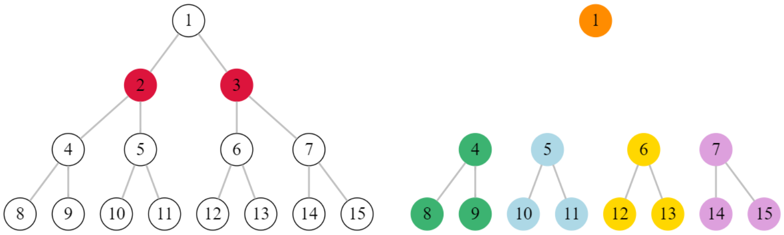


Figure 3: A small graph with 15 nodes. If we delete the second and the third nodes (left), the graph will have 5 connected components (right)

I.4.1 Evolutionary computation method

We present three strategies, which can be used in the initialization phase of any multi-objective algorithm. The first one is based on a depth search algorithm, outlined in Algorithm 11. A Depth First Search (DFS) algorithm is started with a random initial point, and every xth element will be added to the chromosome, where $x = \frac{|V|}{k}$.

The second initialization method is based on the degree distribution of the nodes. The first x nodes with the highest degree are set in the chromosome, and the rest of the $k - x$ nodes are selected randomly to preserve the stochastic nature of the initialization (Algorithm 12).

The third method is based on a random walk. We start the walk in a random node, t is the length of the walk and p_r is the probability to restart the walk. In each step it is decided to continue the walk or to restart. If we failed to walk throw k different nodes the algorithm will restart with an other initial point. In the walk we keep counting how many times a node appeared, the more times, the higher probability to be a gene in the chromosome. The main steps are presented in Algorithm 13.

These initialization strategies can be used in any kind of multi-objective evolutionary algorithm.

Algorithm 11 Depth-first search solution generator

```
Require: $G, k, x$
$start \leftarrow$ SELECT($V$)
$S \leftarrow$ DFS($G, start$)
return $S[:: x]$                                        ▷ Take every $x$th element
```

Algorithm 12 Degree solution generator

```
Require: $G, k, x$
$V' \leftarrow$ SORTED($V$)           ▷ Sort nodes according to their degree in DESC order
$S \leftarrow V'[: x]$                ▷ Take first $x$ nodes with the highest degree
while $|S| < k$ do
    $node \leftarrow$ SELECT($V'$)
    if $node \notin S$ then
        $S \leftarrow S \cup \{node\}$
    end if
end while
SHUFFLE($S$)
return $S$
```

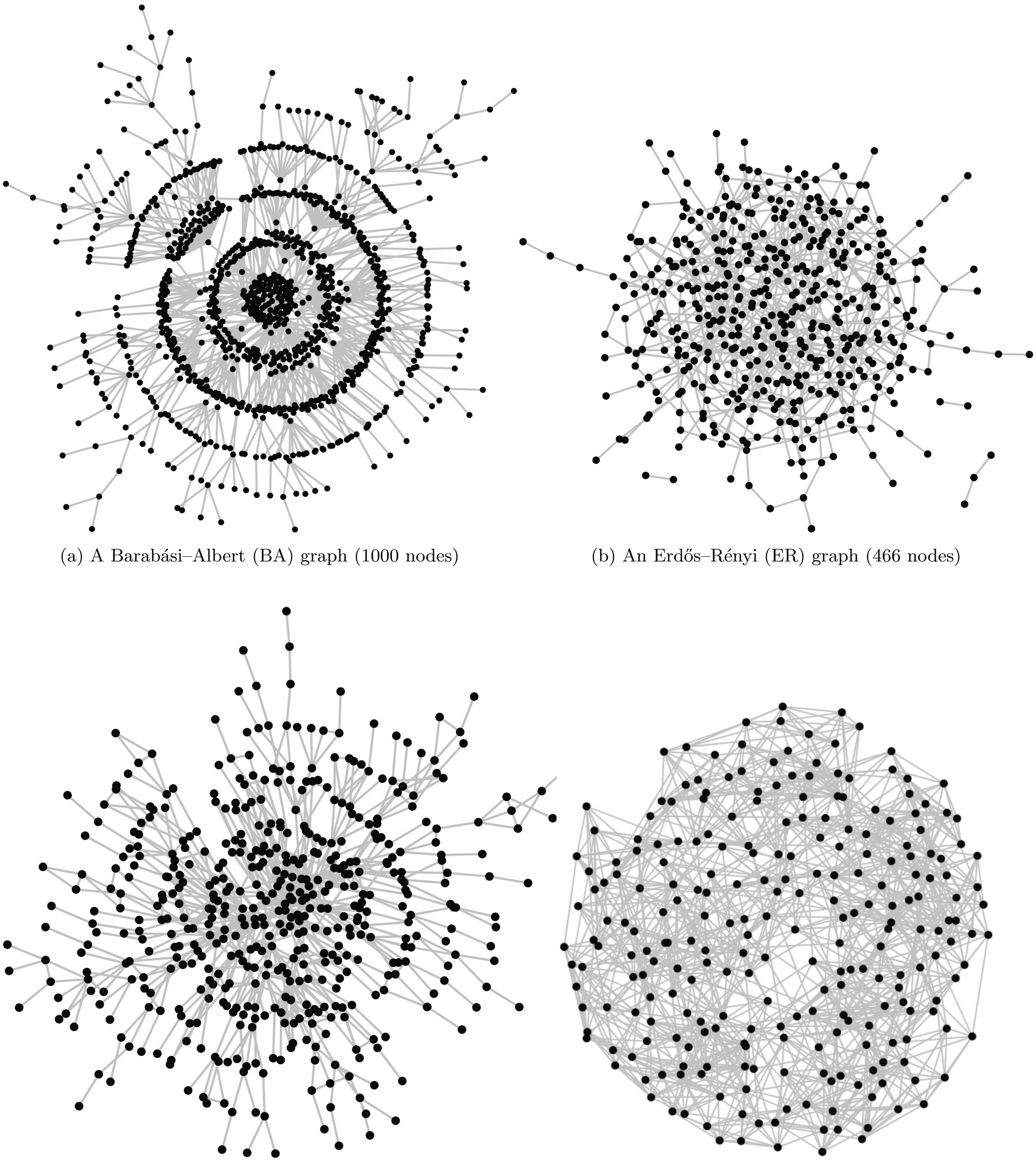
I.4.2 Numerical experiments

Benchmarks We use the graph set proposed in [22]. The benchmark set contains four different type of graphs: Barabási-Albert (BA), Erdős-Rényi (ER), Forest-fire (FF), Watts–Strogatz (WS) graphs. Figure 4 illustrates how the above presented benchmarks visually look.

Table 3 presents some basic properties of the benchmarks used for experiments: number of nodes ($|V|$), number of edges ($|E|$), the number of critical nodes (k), average degree ($\langle d \rangle$), the diameter of the graph (D), density of the graph (ρ), modularity (Q) and average path length (l_G).

Algorithm 13 Random walk solution generator (RWR)

```
Require: $G, k, t, p_r$
visited ← $\emptyset$
while True do
    core ← Select($V$)
    current ← core
    for $i \leftarrow 1, t + 1$ do
        if current ∈ visited then
            visited [current] ← visited [current] + 1
        else
            visited [current] ← 1
        end if
        restart ← Rand Int(1, 100)
        if restart ≤ $p_r$ then
            current ← core
        else
            neighbors ← Neighbors($G$, current)          ▷ Neighbors of the current node
            current ← Select(neighbors)
        end if
    end for
    if |visited| ≥ $k$ then
        break
    else
        visited ← $\emptyset$
    end if
end while
Sort(visited)          ▷ Sort nodes in visited according to visits paid in DESC order
return visited [: $k$]          ▷ Take the first $k$ most visited nodes
```



(a) A Barabási–Albert (BA) graph (1000 nodes)

(b) An Erdős–Rényi (ER) graph (466 nodes)

(c) A Forest-fire (FF) graph (500 nodes)

(d) A Watts–Strogatz (WS) graph (250 nodes)

Figure 4: An illustration of the four types of the synthetic benchmark set

Table 3: Benchmark test graphs and basic properties.

Graph	$\|V\|$	$\|E\|$	k	$\langle d\rangle$	D	ρ	Q	l_G
BA500	500	499	50	1.996	13	0.004	0.886	5.663
BA1000	1000	999	75	1.998	18	0.002	0.910	6.045
BA2500	2500	2499	100	1.999	17	0.001	0.946	6.901
BA5000	5000	4999	150	2.000	24	0.000	0.963	8.380
ER250	235	350	50	2.979	14	0.013	0.603	5.338
ER500	466	700	80	3.004	14	0.006	0.631	5.973
ER1000	941	1400	140	2.976	16	0.003	0.649	6.558
ER2500	2344	3500	200	2.986	16	0.001	0.663	7.516
FF250	250	514	50	4.112	14	0.017	0.638	4.816
FF500	500	828	110	3.312	15	0.007	0.798	6.026
FF1000	1000	1817	150	3.634	20	0.004	0.793	6.173
FF2000	2000	3413	200	3.413	19	0.002	0.880	7.587
WS250	250	1246	70	9.968	6	0.040	0.697	3.327
WS500	500	1496	125	5.984	10	0.012	0.789	5.304
WS1000	1000	4996	200	9.992	7	0.010	0.803	4.444
WS2000	1500	4498	265	5.997	15	0.004	0.872	7.554

Table 4: Parameter settings for NSGA-II

Parameter	Value
Pop size	50
No evaluation	10000
Prob. of crossover	1
Prob. of mutation	0.3

Algorithm For the numerical experiments we used the NSGA-II [6] algorithm within the Platypus[2] framework.

Parameter settings For the numerical experiments parameters of the NSGA-II algorithm are presented in Table 4. The weights of the nodes are set to 1 and W equals to the number of nodes.

Performance evaluation For the performance evaluation we use the hypervolume indicator [24], [25], a popular measure for the multi-objective optimization algorithms. The hypervolume indicator measures the volume of the region of the dominated points in the objective space bounded by a reference point.

I.4.3 Results and discussion

We conducted ten independent runs for each initialization strategy (depth first search, degree based, random walk) and we made comparisons also with random initialization.

Figure 5 presents the obtained Pareto front within a single run.

Table 5 presents the mean values and the standard deviation of the hypervolume indicators. For the reference point we set the nadir point of all unified Pareto fronts. We also conducted a Wilcoxon sign-rank nonparametric test for the hypervolume indicator reported by each method. The Wilcoxon sign rank assesses if there is a significant difference between two sample means. A (*) is used to indicate statistical significance of differences, all initialization strategies which are not statistically different from the best one are marked.

Based on the results, we cannot draw a general conclusion which initialization strategy is better, actually the structure of the graph determines which initialization worth to use (but based on the numerical results, all of them give better results as the random initialization). In the case of Barabási-Albert graphs, which contain hubs, the degree based initialization get the best result. Erdős-Rényi graphs are random graphs, in this case most times the depth first search algorithm seems to be the best one. Forest-fire graphs are also random graphs, replicating how a fire spreads in a forest. In this case, almost all three proposed initialization types gave the same result. The Watts–Strogatz graphs have a dense structure, the best results were provided by the random walk based algorithm.

[2] https://github.com/quaquel/Platypus, last accessed 15/10/2020

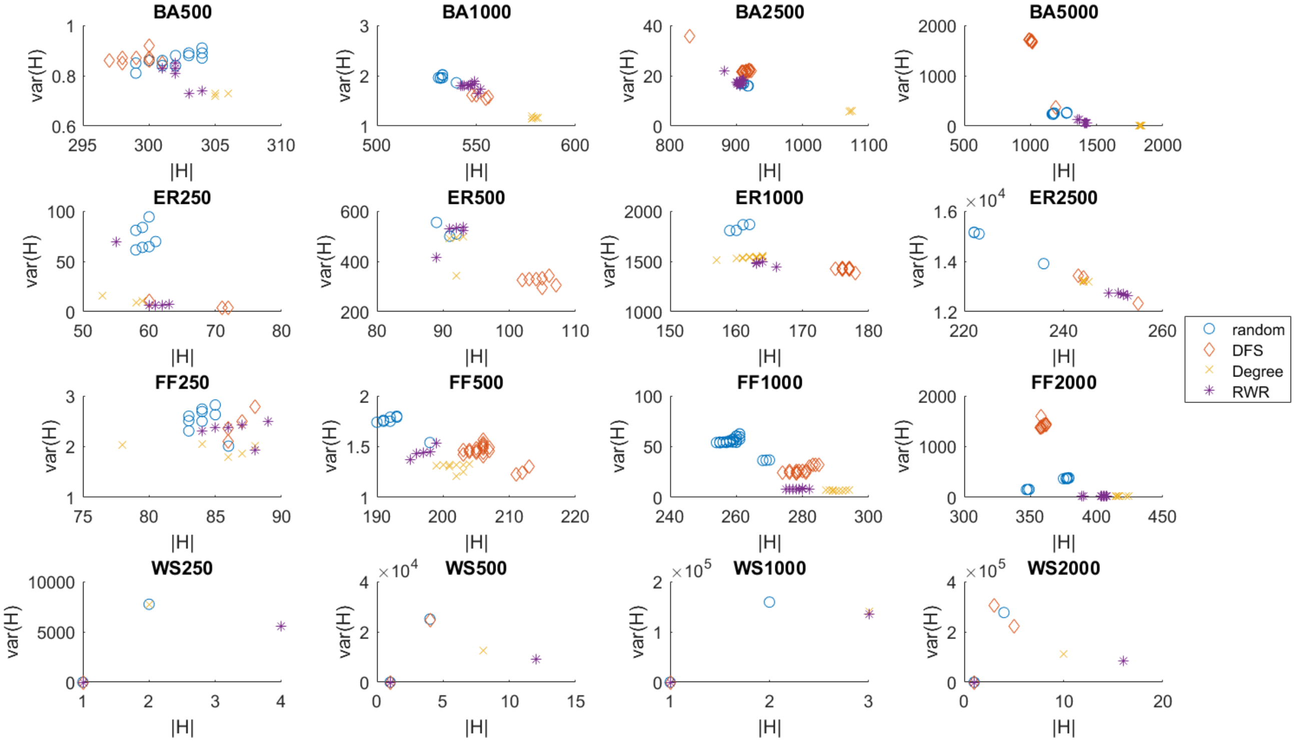


Figure 5: Pareto front obtained in a single run for the 16 benchmark problems

Table 5: Average value ± standard deviation of the hypervolume indicator. A (*) indicates the best results which are not based on a Wilcoxon sign-rank test

Graph	Random	DFS	Degree	RWR
BA500	15.65 ± 4.37	13.03 ± 5.37	$28.05 \pm 2.62^*$	20.72 ± 4.37
BA1000	36.54 ± 21.58	58.38 ± 24.17	$154.27 \pm 10.54^*$	68.21 ± 26.51
BA2500	6138.01 ± 1701.47	2924.91 ± 2828.77	$18607.82 \pm 364.64^*$	8643.83 ± 1993.09
BA5000	1078479.10 ± 319614.94	449069.24 ± 396267.77	$3409301.92 \pm 47950.42^*$	1706543.15 ± 182869.56
ER250	738.68 ± 217.41	$2387.05 \pm 342.45^*$	1162.31 ± 252.49	1204.39 ± 319.17
ER500	1607.75 ± 920.84	$8045.09 \pm 1848.71^*$	2762.01 ± 1572.09	2644.67 ± 919.06
ER1000	1133.40 ± 949.75	$9407.93 \pm 4141.55^*$	$6385.57 \pm 3487.82^*$	5176.12 ± 1499.51
ER2000	39323.23 ± 24560.07	$65449.57 \pm 34152.98^*$	$56123.63 \pm 12683.44^*$	$72259.64 \pm 26198.93^*$
FF250	38.01 ± 6.84	$36.34 \pm 12.26^*$	$41.88 \pm 5.39^*$	$45.61 \pm 7.52^*$
FF500	24.51 ± 8.19	$51.42 \pm 9.68^*$	37.72 ± 4.93	32.98 ± 6.86
FF1000	1402.85 ± 599.70	1718.09 ± 647.33	$3274.96 \pm 387.73^*$	$3194.72 \pm 409.27^*$
FF2000	30560.10 ± 20460.26	16206.67 ± 13027.87	$136548.86 \pm 12378.66^*$	107945.88 ± 12371.82
WS250	8132.90 ± 268.63	9577.11 ± 1390.61	8495.50 ± 636.66	$16193.42 \pm 3306.69^*$
WS500	48526.56 ± 14388.26	35879.19 ± 8263.24	110772.29 ± 31047.79	$203301.59 \pm 30420.58^*$
WS1000	159361.50 ± 503.68	159202.10 ± 0.32	162811.19 ± 11412.20	$337726.99 \pm 90573.11^*$
WS2000	484347.05 ± 140137.66	708776.63 ± 223377.92	1923479.18 ± 351573.85	$5058526.43 \pm 1741808.94^*$

Table 6: Results of the NSGAII for BOCNDP using Nash and Berge generative relations.

		$\|H\|$		$var(H)$			$\|H\|$		$var(H)$	
Graph	Algorithm	μ	σ	μ	σ	Algorithm	μ	σ	μ	σ
BA500	N-NSGAII	111.40	14.49	313.28	130.22	B-NSGAII	121.80	15.01	209.53	81.78
ER500		39.10	1.91	2,742.85	192.65		41.00	1.89	2,623.55	170.18
BA1000		195.40	19.45	199.53	56.67		217.60	18.46	131.34	33.66
ER1000		60.60	2.37	8,330.21	418.18		59.80	2.39	8,427.53	418.45
FF250		38.60	4.25	395.91	57.94		38.90	2.77	377.91	60.25
WS250		1.00	0.00	0.00	0.00		1.00	0.00	0.00	0.00
FF1000		90.90	4.58	4,706.83	475.98		90.60	4.53	4,405.25	253.48
WS1000		1.00	0.00	0.00	0.00		1.00	0.00	0.00	0.00

I.4.4 Game theory-based approach

We replaced the Pareto domination concept with the presented Nash domination and also with the Berge domination [10]. The main idea behind this is to obtain a single solution to the problem. Table 6 presents the obtained results. The used algorithm is the NSGA-II.

References

[1] B. Addis, M. Di Summa, and A. Grosso. Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. *Discrete Applied Mathematics*, 161(16-17):2349–2360, 2013.

[2] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia. A general evolutionary framework for different classes of critical node problems. *Engineering Applications of Artificial Intelligence*, 55:128–145, 2016.

[3] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

[4] A. Arulselvan, C. W. Commander, O. Shylo, and P. M. Pardalos. Cardinality-constrained critical node detection problem. In *Performance models and risk management in communications systems*, pages 79–91. Springer, 2011.

[5] S. Boettcher and A. G. Percus. Optimization with Extremal Dynamics. *Physical Review Letters*, 86:5211–5214, 2001.

[6] K. D. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[7] M. Di Summa, A. Grosso, and M. Locatelli. Branch and cut algorithms for detecting critical nodes in undirected graphs. *Computational Optimization and Applications*, 53(3):649–680, 2012.

[8] N. Fan and P. M. Pardalos. Robust optimization of graph partitioning and critical node detection in analyzing networks. In *International Conference on Combinatorial Optimization and Applications*, pages 170–183. Springer, 2010.

[9] S. Fortunato. Community detection in graphs. *Physics Reports*, 486:75 – 174, 2010.

[10] N. Gaskó, D. Dumitrescu, and R. I. Lung. Evolutionary detection of Berge and Nash equilibria. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2011)*, volume 387 of *Studies in Computational Intelligence*, pages 149–158. Springer Berlin / Heidelberg, 2012.

[11] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[12] C. J. Kuhlman, V. A. Kumar, M. V. Marathe, S. Ravi, and D. J. Rosenkrantz. Finding critical nodes for inhibiting diffusion of complex contagions in social networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 111–127. Springer, 2010.

[13] M. Lalou, M. A. Tahraoui, and H. Kheddouci. Component-cardinality-constrained critical node problem in graphs. *Discrete Applied Mathematics*, 210:150–163, 2016.

[14] M. Lalou, M. A. Tahraoui, and H. Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117, 2018.

[15] V. Latora, V. Nicosia, and G. Russo. *Complex networks: principles, methods and applications.* Cambridge University Press, 2017.

[16] J. Li, P. M. Pardalos, B. Xin, and J. Chen. The bi-objective critical node detection problem with minimum pairwise connectivity and cost: Theory and algorithms. *Soft Computing*, 23(23):12729–12744, 2019.

[17] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[18] X. Liu, Z. Hong, J. Liu, Y. Lin, A. Rodríguez-Patón, Q. Zou, and X. Zeng. Computational methods for identifying the critical nodes in biological networks. *Briefings in bioinformatics*, 21(2):486–497, 2020.

[19] R. I. Lung and D. Dumitrescu. Computing nash equilibria by means of evolutionary computation. *Int. J. of Computers, Communications & Control*, 3(suppl. issue):364–368, 2008.

[20] R. I. Lung, T. D. Mihoc, and D. Dumitrescu. Nash Extremal Optimization and Large Cournot Games. In *Nature Inspired Cooperative Strategies for Optimization, {NICSO} 2011, Cluj-Napoca, Romania, October 20-22, 2011*, pages 195–203, 2011.

[21] Z. Tao, F. Zhongqian, and W. Binghong. Epidemic dynamics on complex networks. *Progress in Natural Science*, 16(5):452–457, 2006.

[22] M. Ventresca. Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem. *Computers & Operations Research*, 39(11):2763–2775, 2012.

[23] M. Ventresca, K. R. Harrison, and B. M. Ombuki-Berman. The bi-objective critical node detection problem. *European Journal of Operational Research*, 265(3):895–908, 2018.

[24] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[25] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.