







A Simple Genetic Algorithm for the Critical Node Detection Problem

Mihai-Alexandru Suciu^{}, Noémi Gaskó^{}, Tamás Képes,
and Rodica Ioana Lung^{^{}}

Centre for the Study of Complexity, Babeş-Bolyai University, Cluj-Napoca, Romania
{mihai.suciu,noemi.gasko,tamas.kepes,rodica.lung}@ubbcluj.ro
<http://csc.centre.ubbcluj.ro/>

Abstract. The critical node detection problem describes a class of graph problems that involves identifying sets of nodes that influence a given graph metric. One variant of this problem is to find the nodes that - when removed from the graph - maximize the number of connected components in the remaining graph. This is an example of a practical problem with multiple real-world applications in epidemic control, immunization strategies, social networks, biology, etc. This paper proposes the use of a simple GA to identify the set of the critical nodes of the problem without designing special problem specific variation operators. Problem specific information is used only in the fitness function and the constraint handling technique. We show that this simple approach performs as well as state-of-art methods.

Keywords: Critical node detection · Genetic algorithm · Complex networks

1 Introduction

Nodes in a network can have different importance with respect to different network measures and behavior. Finding these nodes, called critical nodes, is an essential computational task. Critical nodes can be approached also from the general node deletion problem [14], which is a large class of problem composed of several problems, such as the vertex separator problem, the minimum vertex cover problem, the critical node detection problem, etc. Recently, the critical node detection problem (CDNP) gained attention due to its large applicability. A very important class of the critical node detection problem is to identify the set of nodes of a maximal size to remove from the graph in order to maximize the number of connected components. Applications of this problem can be found in epidemic control and immunization strategies, social networks, biology, telecommunications, etc.

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS - UEFISCDI, project number PN-III-P1-1.1-TE-2019-1633.

In general, the critical node detection problem consists in finding a set of nodes in a given graph $G = (V, E)$, which deleted maximally degrades the graph according to a given measure σ . CDNP is a central problem in network analysis with applications in several research fields, such as biology [2], network vulnerability [6], social network analysis [3], etc. Regarding the measure σ several studies focus on network centrality measures, such as betweenness centrality, closeness centrality, page rank [11, 16].

Although several variants of the CDNP exist, only a few of them deal with computational methods for the variant consisting of removing k nodes in order to maximize the number of remaining components. The main goal of this paper is to approach this problem using a genetic algorithm with minimal problem specific adaptations. The choice of a genetic algorithm came first due to the natural binary encoding of an individual, but this is not the only reason we made it: we believe that it is important to explore different methods and paths and not constrain ourselves to assuming that one method may not work on a certain problem because it has not been tested on it. This is also related to the choice of operators: if there is not need for specific operators that use domain knowledge, we should not use them and keep the approach as general and as flexible as possible.

The rest of the paper is organized as follows: the next section presents the problem and reviews some existing approaches. The third section describes the proposed genetic algorithm. In the fourth section numerical experiments considering synthetic and real world networks are used to compare our results with the existing ones. The article ends with conclusions and further work.

2 Related Work

Many variants of the critical node detection problem are studied in the literature, among which we mention: minimizing the pairwise connectivity by deleting k nodes (this variant is the most studied in the literature), minimizing the largest component size by deleting k nodes, bound the pairwise connectivity to a given threshold by deleting the minimal set of nodes, etc. A recent survey of the problem can be found in [13].

There are several ways to classify the critical node detection problem (CNDP). In [21] the two types variant is adopted: *CNDP type 1* problems aim to minimize the network connectivity maintaining the number of removed nodes under a given threshold and *CNDP type 2* problems in which the goal is to minimize the number of nodes that are removed such that the network connectivity reaches a given threshold. The type of connectivity measure used depends on the envisaged application, effect or the type of network. Applications are multiple as the CNDP is related to network sustainability and vulnerability [21]. Many practical approaches are devised for wireless sensor networks [7, 8, 18].

In [21] an exact algorithm for the problem considering the largest connected component is proposed. The k -vertex cut problem, consisting in finding the minimum weight subset whose removal disconnects the graph in at least k compo-

nents is studied in [9]. Component-Cardinality-Constrained Critical Node Problem (3C-CNP) is approached in [12]. A bi-objective design is presented in [25]. As far as the type of networks, weighted networks are studied for example in [5] and directed graphs in [19].

In [1] the two types of CNDP problems are studied in three versions, among which also *kMaxComp*, the problem of removing a set of maximum k nodes to maximize the number of connected components in the remaining graph. This is one of the less studied CNDP variants, proven to be NP-hard [24]. In [24] a Mixed Integer linear programming approach is presented, [27] present a general integer programming framework. For a special class of graphs (trees and series-parallel graphs) a dynamic programming approach is presented [23]. In [1] a genetic algorithm is designed to solve the problem. The proposed genetic algorithm incorporates in the fitness function a penalization of solutions that are too close to the best solutions, combines a greedy strategy with variation operators and employs a local search mechanism at the end in order to refine solutions.

In this paper we focus on the problem $CDNP_a^3$, denoted here as *kMaxComp*, introduced in [23,24]. The $CDNP_a^3$ is by itself an interesting problem to be studied, with many possible applications. It has received less attention because it does not impose any conditions on the connected components. The problem consists in removing a maximum of k nodes such as the number of remaining components to be maximal. Formally, if S denotes the set of the deleted nodes, and $\mathcal{H}(G[V \setminus S])$ denotes the set of the maximal component of graph G without the set of nodes S , the optimization problem consists in

$$\max |\mathcal{H}(G[V \setminus S])|, \text{ such that } |S| \leq k, \quad (1)$$

where $|A|$ denotes the cardinality of set A .

3 Maximum Components GA (MaxC-GA)

The goal of this work is to solve the *kMaxComp* problem by using a minimum number of problem specific information during the search. Because we search for a set of nodes from a network out of which some will be included in the critical set S and some not, a binary encoding of an individual of length $N = |V|$ is natural, making a genetic algorithm the first choice in trying to approach this problem. We call this algorithm Maximum Components GA. MaxC-GA is outlined in Algorithm 1. MaxC-GA is a simple approach for the $CDNP_3a$ problem, that combines a standard GA with a constraint method based on the marginal contribution of a node to the fitness of an individual, concept borrowed from game theory, where such marginal contributions are used to evaluate the contribution of a player to the value of a coalition when computing the Shapley value [22].

Encoding. An individual has length N equal to the number of nodes in the network. The value 1 on position i indicates that node i is included in S .

Algorithm 1. MaxC-GA outline

```

Initialize population  $P$  of size  $p_{size}$  at random.
for a number of generations do
     $P =$  Select  $p_{size}$  individuals for variation;
    Offspring= variation operators on  $P$ ;
    Correct and Evaluate Offspring;
     $P =$  offspring;
end for

```

Variation Operators. Two point crossover and flip-bit mutation are used.

Selection. Tournament selection is used for selection for recombination and mutation.

Fitness Function. The fitness of an individual is computed as the number of connected components the removal of its nodes with value 1 yields. Thus, if individual x encodes the critical set S_x then the fitness $f(x)$ of x is computed as

$$f(x) = |\mathcal{H}(G[V \setminus S_x])|. \quad (2)$$

Constraint Handling. In order to ensure that the size of the corresponding set S does not exceed k , before evaluation each individual is constrained to have only k nodes with value 1 by removing the nodes with the lowest marginal contribution to the fitness of the individuals from S . The marginal contribution of a node to the fitness of the individual is computed as the difference between the fitness of the individual and the fitness of the individual with the node removed from its corresponding set S of critical nodes. For a node i with value 1 in individual x with corresponding critical set S_x the marginal contribution of node i to the fitness of x denoted by $u_i(x)$ is:

$$u_i(x) = f(x) - |\mathcal{H}(G[V \setminus \{S_x \setminus \{i\}])|,$$

where $f(x)$ is the fitness defined in Eq. (2).

Parameters. MaxC-GA is a standard GA, and uses typical GA parameters: maximum number of generations, crossover and mutation probabilities, probability to mutate a bit, and tournament size. The effect of these parameters on the search results of a GA has been widely documented [15].

4 Numerical Experiments

The behavior of MaxC-GA is illustrated by using several benchmarks and comparing results with best known found in the literature for this problem.

Table 1. Synthetic benchmark test graphs and basic properties.

Graph	$ V $	$ E $	k	$\langle d \rangle$	ρ	l_G
BA1000	1000	999	75	1.998	0.002	6.045
ER466	466	700	80	3.004	0.006	5.973
FF500	500	828	110	3.312	0.007	6.026
WS500	500	1496	125	5.984	0.012	5.304

Benchmarks. A set of synthetic benchmarks¹ was proposed in [26]. The benchmark set contains four different type of graphs: Barabási-Albert (BA), Erdős-Rényi (ER), Forest-fire (FF), Watts–Strogatz (WS) graphs. BA graphs are scale free networks, ER graphs are random networks, FF graphs simulate how fire spreads through a forest, WS graphs are small world graphs with a dense structure.

Table 1 presents some basic measures of the benchmarks used for numerical experiments here: number of nodes ($|V|$), number of edges ($|E|$), average degree ($\langle d \rangle$), density of the graph (ρ), and average path length (l_G). In a similar manner, real networks are described in Table 2 with a reference added for each network.

Table 2. Real-world graphs and basic properties.

Graph	$ V $	$ E $	k	$\langle d \rangle$	ρ	l_G	Ref.
Bovine	121	190	3	3.140	0.026	2.861	[20]
Circuit	252	399	25	3.167	0.012	5.806	[17]
EColi	328	456	15	2.780	0.008	4.834	[28]
HumanDis	516	1188	52	4.605	0.008	6.509	[10]
TrainsRome	255	272	26	2.133	0.008	43.496	[4]

Parameter Settings. Several parameter setting are tested: population size set to 25 and 50, maximum number of generations 500, crossover probability 0, 0.5, 0.8, and 1, and mutation rate 0, 0.01, 0.02, 0.03, 0.04, and 0.05.

Results and Discussion. MaxC-GA is compared with three algorithms described in [1]: two greedy algorithms, the first one, G_1 based on node deletion from the candidate critical node set, and the second one, G_2 , based on the node addition to the candidate critical node set and a genetic algorithm from an evolutionary algorithm framework using greedy rules (denoted by GA). The genetic algorithm uses a specific fitness function that combines the number of connected components determined by the interval with previous search information, problem specific variation operators and a specific designed local search technique.

¹ downloaded from <http://individual.utoronto.ca/mventresca/cnd.html>, last accessed 05.09.2020.

Table 3. Maximum fitness values for the tested problems. The average over 10 runs is presented for MaxC-GA.

Graph	MaxC-GA	GA	G_1	G_2
ER466	110.0	110	99	105
BA1000	590.0	590	590	590
FF500	215.0	215	214	214
WS500	44.0	44	25	18
Bovine	77.0	77	77	77
Circuit	32.0	31	30	29
EColi	169.0	169	169	168
HumanDis	148.0	148	147	148
TrainsRome	31.0	31	30	31

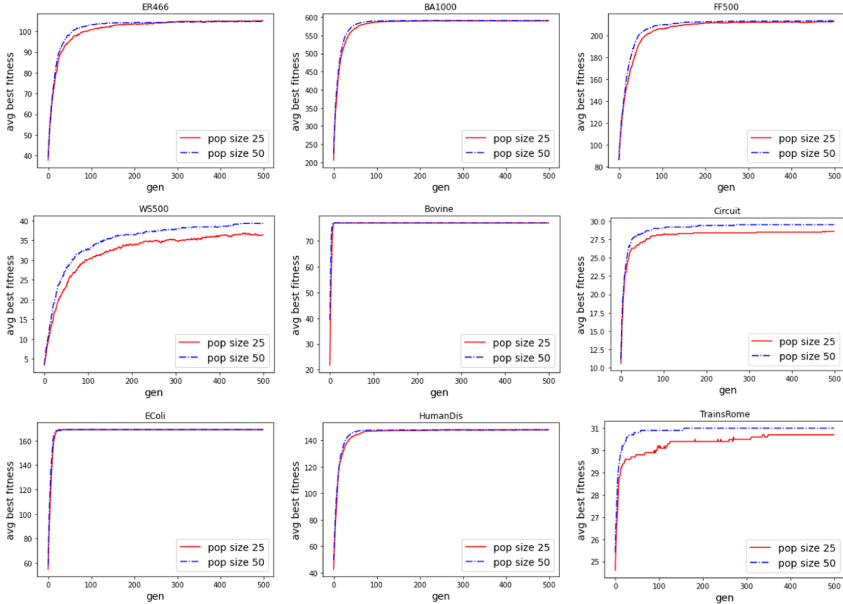


Fig. 1. Search evolution of MaxC-GA for the benchmarks, average best fitness for a population size of 25 and 50 over 10 runs.

Since the problem has been less addressed, we only have one approach based on GAs to compare with, and those results represent only one run. Results presented in the paper are preliminary and promising, supporting the idea that this approach may be extended for larger data sets.

As results presented in [1] include only the maximum number of connected components in one run, therefore statistical comparisons with results reported there are not possible. Table 3 includes these results as well best results reported

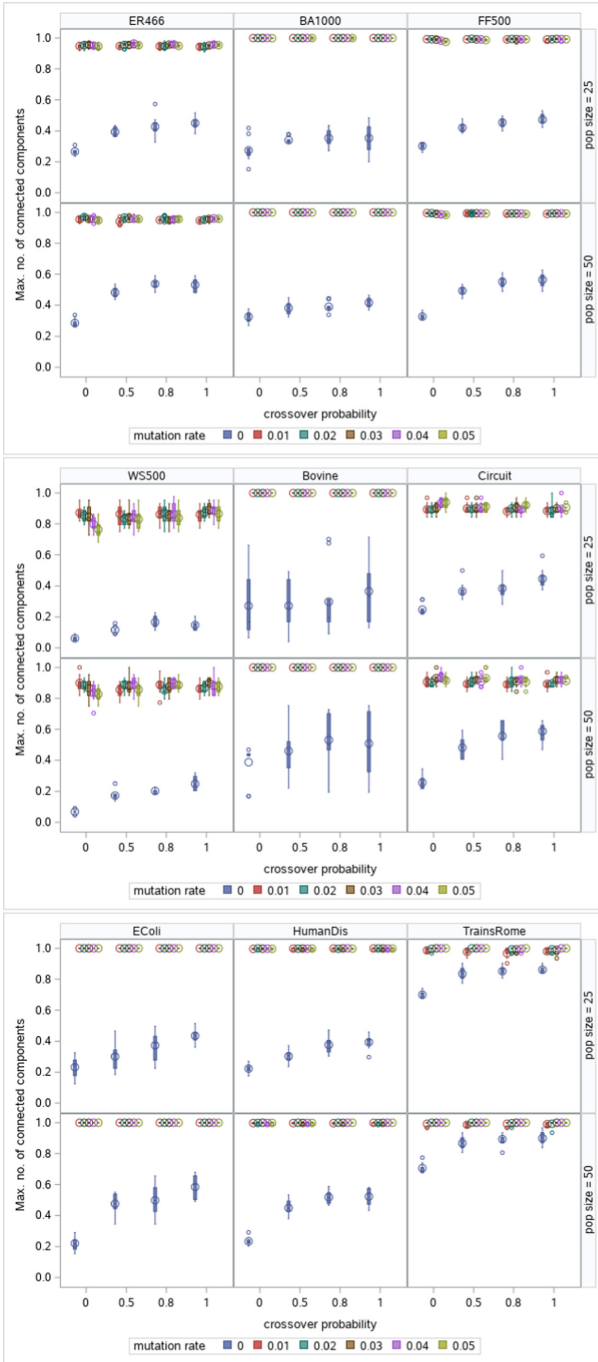


Fig. 2. Box plots presenting results reported by MaxC-GA for the nine benchmarks and different parameter values.

by MaxC-GA. Results reported by MaxC-GA using different parameter settings are illustrated in Fig. 2. Furthermore, Fig. 1 illustrates the evolution of the search of MaxC-GA (average best solutions over 10 runs). We find that the evolution is steady, faster for a larger population size, and that MaxC-GA is capable to find and maintain the optimal solution. Because the behavior of MaxC-GA under different parameter settings is typical for that of a GA, with respect to convergence we have presented only graphs showing that it is capable to detect and maintain the optimal solution during one run. In all other ways it behaves as expected: a larger population size leads to an earlier convergence at a higher computational cost and a small population will eventually converge.

The effect of various parameter settings presented in Fig. 2 as boxplots of the ratio of maximum fitness values reported in 10 runs for each parameter setting and best known result for the benchmark (in order to keep all values between 0 and 1). We find that the algorithm is robust with respect to variation of parameters, with the notable exception that mutation plays an important role in the search, as setting the mutation rate to 0 significantly decreases the performance of the algorithm.

5 Conclusions

The critical node detection problem is approached with MaxC-GA, a simple genetic algorithm that uses a node fitness based on marginal contributions for constraint handling. Numerical results show that this approach is as effective as other, more complex, using more problems specific information.

These results may also be used to advocate for the use of minimal problem specific information in designing new evolutionary algorithms for real-world applications. Overusing specific problem information decreases the adaptability of the presented method, as practitioners will rarely try to adapt an existing algorithm presented in literature to a slightly different problem, mainly because the stochastic nature of these approaches does not guarantee direct portability to a different problem.

References

1. Aringhieri, R., Grosso, A., Hosteins, P., Scatamacchia, R.: A general evolutionary framework for different classes of critical node problems. *Eng. Appl. Artif. Intell.* **55**, 128–145 (2016)
2. Boginski, V., Commander, C.W.: Identifying critical nodes in protein-protein interaction networks. In: *Clustering Challenges in Biological Networks*, pp. 153–167. World Scientific (2009)
3. Borgatti, S.P.: Identifying sets of key players in a social network. *Comput. Math. Organ. Theory* **12**(1), 21–34 (2006). <https://doi.org/10.1007/s10588-006-7084-x>
4. Cacchiani, V., Caprara, A., Toth, P.: Scheduling extra freight trains on railway networks. *Transp. Res. Part B: Methodol.* **44**(2), 215–231 (2010)

5. Chen, W., Jiang, M., Jiang, C., Zhang, J.: Critical node detection problem for complex network in undirected weighted networks. *Phys. A: Stat. Mech. Appl.* **538** (2020). <https://doi.org/10.1016/j.physa.2019.122862>
6. Cohen, R., Erez, K., Ben-Avraham, D., Havlin, S.: Resilience of the internet to random breakdowns. *Phys. Rev. Lett.* **85**(21), 4626 (2000)
7. Dagdeviren, O., Akram, V.: An energy-efficient distributed cut vertex detection algorithm for wireless sensor networks. *Comput. J.* **57**(12), 1852–1869 (2013). <https://doi.org/10.1093/comjnl/bxt128>
8. Dagdeviren, O., Akram, V., Tavli, B., Yildiz, H., Atilgan, C.: Distributed detection of critical nodes in wireless sensor networks using connected dominating set (2017). <https://doi.org/10.1109/ICSENS.2016.7808815>
9. Furini, F., Ljubić, I., Malaguti, E., Paronuzzi, P.: On integer and bilevel formulations for the k -vertex cut problem. *Math. Program. Comput.* **12**(2), 133–164 (2019). <https://doi.org/10.1007/s12532-019-00167-1>
10. Goh, K.I., Cusick, M.E., Valle, D., Childs, B., Vidal, M., Barabási, A.L.: The human disease network. *Proc. Natl. Acad. Sci.* **104**(21), 8685–8690 (2007)
11. Iyer, S., Killingback, T., Sundaram, B., Wang, Z.: Attack robustness and centrality of complex networks. *PloS One* **8**(4), e59613 (2013)
12. Lalou, M., Tahraoui, M., Kheddouci, H.: Component-cardinality-constrained critical node problem in graphs. *Discrete Appl. Math.* **210**, 150–163 (2016). <https://doi.org/10.1016/j.dam.2015.01.043>
13. Lalou, M., Tahraoui, M., Kheddouci, H.: The critical node detection problem in networks: a survey. *Comput. Sci. Rev.* **28**, 92–117 (2018). <https://doi.org/10.1016/j.cosrev.2018.02.002>
14. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.* **20**(2), 219–230 (1980)
15. Lobo, F., Lima, C.F., Michalewicz, Z.: *Parameter Setting in Evolutionary Algorithms*, vol. 54. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-3-540-69432-8>
16. Lozano, M., García-Martínez, C., Rodríguez, F.J., Trujillo, H.M.: Optimizing network attacks by artificial bee colony. *Inf. Sci.* **377**, 30–50 (2017)
17. Milo, R., et al.: Superfamilies of evolved and designed networks. *Science* **303**(5663), 1538–1542 (2004)
18. Min, S., Jiandong, L., Yan, S.: Critical nodes detection in mobile ad hoc network, vol. 2, pp. 336–340 (2006). <https://doi.org/10.1109/AINA.2006.136>
19. Paudel, N., Georgiadis, L., Italiano, G.: Computing critical nodes in directed graphs. *ACM J. Exp. Algorithmics* **23** (2018). <https://doi.org/10.1145/3228332>
20. Reimand, J., Tooming, L., Peterson, H., Adler, P., Vilo, J.: GraphWeb: mining heterogeneous biological networks for gene modules with functional significance. *Nucleic Acids Res.* **36**, 452–459 (2008)
21. Rezaei, J., Zare-Mirakabad, F., MirHassani, S., Marashi, S.A.: EIA-CNDP: an exact iterative algorithm for critical node detection problem. *Comput. Oper. Res.* **127** (2021). <https://doi.org/10.1016/j.cor.2020.105138>
22. Shapley, L.S.: 17. A Value for n -Person Games, pp. 307–317. Princeton University Press (1953). DOIurl10.1515/9781400881970-018
23. Shen, S., Smith, J.C.: Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks* **60**(2), 103–119 (2012)
24. Shen, S., Smith, J.C., Goli, R.: Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optim.* **9**(3), 172–188 (2012)

25. Ventresca, M., Harrison, K., Ombuki-Berman, B.: The bi-objective critical node detection problem. *Eur. J. Oper. Res.* **265**(3), 895–908 (2018). <https://doi.org/10.1016/j.ejor.2017.08.053>
26. Ventresca, M.: Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem. *Comput. Oper. Res.* **39**(11), 2763–2775 (2012)
27. Veremyev, A., Prokopyev, O.A., Pasiliao, E.L.: An integer programming framework for critical elements detection in graphs. *J. Comb. Optim.* **28**(1), 233–273 (2014). <https://doi.org/10.1007/s10878-014-9730-4>
28. Yang, R., Huang, L., Lai, Y.C.: Selectivity-based spreading dynamics on complex networks. *Phys. Rev. e* **78**(2), 026111 (2008)