A Fast Greedy Algorithm for the Critical Node Detection Problem

Mario Ventresca[1](✉) and Dionne Aleman[2]

[1] School of Industrial Engineering, Purdue University, West Lafayette, USA
mventresca@purdue.edu
[2] Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada

Abstract. The critical node detection problem (CNDP) aims to fragment a graph $G = (V, E)$ by removing a set of vertices R with cardinality $|R| \leq K$ such that the residual graph has minimum pairwise connectivity. Algorithms that are capable of finding R in graphs with many thousands or millions of vertices are needed since existing approaches require significant computational cost and subsequently are useful for only very small network instances. An efficient method for evaluating the impact of removing any $v \in V$ on the CNDP objective function within reasonable time and space complexity is then necessary. In this paper we propose a depth-first search solution to this problem that requires $\mathcal{O}(|V| + |E|)$ complexity, and employ the method in a greedy algorithm for quickly identifying R in large networks. We evaluate the results using six real-world benchmark problems. The proposed algorithm can be easily extended to vertex and edge-weighted variants of the critical vertex detection problem.

1 Introduction

Detecting important or critical vertices in a graph has many important applications. Depending on the context, we may wish to promote or mitigate the diffusive process acting on the network by identifying and utilizing critical nodes. In the context of promoting a spreading process, such as advertisements or health warnings, the notion of "critical" refers to the identification of individuals who are most likely to be influential spreaders and maximally allow the information spread through the network. In such cases the selected individuals may be targeted for demonstrations, promotions or invited to public events. For mitigation contexts, such as stopping computer virus spread or for construction of stable power delivery networks, the identified vertices are those whose removal from the graph will maximally mitigate the threat, hopefully alleviating it entirely.

Specific definitions of what a critical node is have been investigated, including junctions in cell-signalling or protein-protein networks [4], highly influential individuals [12], smart grid vulnerability [17], targeted vaccination for pandemic prevention [3,23] or keys to decipher brain functionality [11]. In some contexts an accurate mathematical definition for a critical vertex, particularly for highly

Z. Zhang et al. (Eds.): COCOA 2014, LNCS 8881, pp. 603–612, 2014.
DOI: 10.1007/978-3-319-12691-3_45

complex systems such as the brain [20], may not yet exist. In any event, it is important to note that both promotion and mitigation can often be defined in a mathematically similar manner.

We concentrate in this paper on the critical node detection problem (CNDP) [2]. Given graph $G = (V, E)$ where $|V| = n$ and $|E| = m$, ascertain (a typically small) subset of vertices, $R \subseteq V$, $|R| \leq K$, whose removal leaves the graph with minimum pairwise connectivity, i.e.,

$$\arg\min_{R \subseteq V} \sum_{\mathcal{C}_i} \binom{|\mathcal{C}_i|}{2} \tag{1}$$

where the sum is over all connected components $\mathcal{C}_i$ of the residual graph $G \backslash R$, and $|\mathcal{C}_i|$ indicates the number of vertices in component i. The optimal network is therefore one that is maximally fragmented, and simultaneously minimizes the variance amongst the component sizes. That is, the residual network $G \backslash R$ will contain a relatively large set of connected components each containing a similar number of vertices. The problem has been shown to be $\mathcal{NP}$-hard [2,9].

1.1 Related Work

The case where G is a tree structure has been examined and proven to be $\mathcal{NP}$-complete when considering non-unit edge costs [8]. A polynomial time dynamic programming algorithm with worst-case complexity $\mathcal{O}(n^3K^2)$ for solving the problem with unit edge costs was also provided and applied to variants of the CNDP [1]. In [7], an integer linear programming model with a non-polynomial number of constraints is given and branch-and-cut algorithms are proposed. A reformulation of the CNDP that requires $\Theta(n^2)$ constraints was recently shown and optimal solutions for small networks were ascertained [27,28].

Heuristic solutions lacking provable approximation bounds have also been investigated, but computation time for very large networks can be further improved. The CNDP work of [2] utilizes a solution to the maximum-independent set problem as a starting point for local search, repeating the process until a desired termination criteria is reached. The algorithm is tested on a limited number of network structures with promising results. Two stochastic search algorithms are employed in [22] that permit solutions to significantly larger networks (up to a few thousand vertices) to be solved within reasonable time and without significant resources. These simulated annealing and population-based incremental learning algorithms are shown to yield quality results. Randomized rounding-based algorithms have been also proposed in [24,26], but without approximation bounds (although, an instance-specific bound was derived). A $\mathcal{O}(\log n \log\log n)$ pseudo-approximation algorithm was proposed in [9].

Among the many problems that have been defined, some of the most similar include the following. The minimum contamination problem [14] aims to minimize the expected size of contamination by removing a set of edges of at most a given cardinality. A variant of the problem is also proposed with the goal of minimizing the proportion of vertices in the largest resulting network. A bi-criteria algorithm is given that is able to achieve an $\mathcal{O}\left(1+\epsilon, \frac{1+\epsilon}{\epsilon}(\log n)\right)$ approximation. In [6], a game-theoretic analysis is conducted that requires a solution to a generalization of the sum-of-squares partitioning problem [3]. Exact methods for link-based vulnerability assessment using edge-disruptors have also recently been investigated [10].

The remainder of this paper is organized as follows. The proposed greedy algorithm and its motivation are provided in Sect. 2. Section 3 provide brief experimentation using six real-world networks. Conclusions are then presented in Sect. 4.

2 The Proposed Algorithm

In this section we propose a greedy algorithm for the CNDP. To overcome computational bottlenecks for larger networks a linear-time algorithm for evaluating the CNDP objective is given and a priority-queue based implementation to yield significant practical performance increases is briefly described. Our $\mathcal{O}(K(n+m))$ algorithm is similar to the $\mathcal{O}(Kn^2)$ approach used for the maximum cascading algorithm in [17]. Both algorithms are based on Tarjan [21].

For large networks with many thousands or millions of vertices (and edges) a computationally efficient approach to minimizing the CNDP objective is required. Selecting critical subset R from a single observation of the network is easily deceived due to the influence of cut vertices, as indicated in Lemma 1 [2]. That is, selecting R in a sequential fashion will allow for the discovery of a set that is more likely to fragment the network by detecting cut vertices that are not obvious unless a sequential approach is taken.

Lemma 1. *Let M_1 and M_2 be two sets of partitions obtained by deleting D_1 and D_2 sets of vertices from graph $G = (V, E)$, where $|D_1| = |D_2| = K$. Let L_1 and L_2 be the number of components in M_1 and M_2 respectively and $L_1 \geq L_2$. If $\mathcal{C}_h = \mathcal{C}_\ell$, $\forall h, \ell \in M_1$, then we obtain a better objective function value by deleting the set D_1.*

Thus, we propose a sequential greedy approach as shown in Algorithm 1. At each iteration the vertex whose removal will have the largest decrease on the objective function (Eq. 1) is selected for removal and added to set R, where $f()$ computes the objective value. Computation of line 3 is a bottleneck to solving large-scale problems. Naively, it implies removal of each $v \in V \setminus R$ and re-evaluation of the objective function.

Here we provide an $\mathcal{O}(K(n+m))$ algorithm based on a modified depth-first search (DFS). The iterative (versus recursive) algorithm of DFS should be used as the framework because sufficiently large networks will quickly encounter stack overflow errors during the search. Performing a DFS on G will construct an equivalent representation as a DFS-tree $DFS(v)$ rooted at arbitrary v. Figure 1 provides an example of the conversion between an original graph to a DFS tree rooted at $v = 0$. Our solution is based on the DFS-tree viewpoint.

Algorithm 1. High-level pseudocode for GREEDY-CNDP.

```
Require: $K > 1$ upper limit on the number of vertices to remove
1: Let $R = \emptyset$ be a set of removed vertices.
2: repeat
3:     select $v^* = \arg\min_{v \in V} f(G \setminus \{v^*\})$
4:     remove $v^*$ from $G$, i.e., $G = G \setminus \{v^*\}$
5:     $R = R \cup \{v^*\}$
6: until $|R| = K$ or $|E| = 0$
```

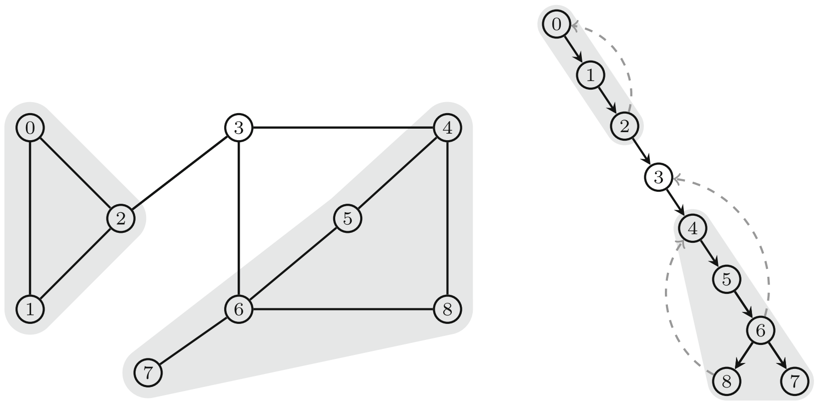

Fig. 1. Equivalent original graph and DFS-tree. Back-edges are indicated as dashed arrows in DFS(0). The shaded areas correspond to resulting connected components if vertex $v = 3$ is removed from the graph. A traditional application of DFS is to detect cut vertices (i.e., articulation points), which forms the basis for our approach.

Observation 1. *Ignoring back-edges, a leaf vertex v has no children. Hence, the subtree rooted at v contains a single vertex whose deletion will not create any new connected components.*

Observation 2. *Let $\delta(v)$ be the set of children vertices of v in the DFS-tree, ignoring back edges. Then, the total number of descendant vertices can be recursively defined as*

$$s(v) = \sum_{w \in \delta(v)} \begin{cases} s(w), & \text{if } w \text{ is an internal or root vertex} \\ 1, & \text{if } w \text{ is a leaf} \end{cases} \tag{2}$$

Observation 3. *Ignoring back-edges, each internal vertex v will either be a cut vertex or not. Removing any $v \in V$ will result in an updated objective value but if v^* is a cut vertex, then the residual graph $G(v \setminus \{v^*\})$ will contain a set of new DFS subtrees $T(v)$. Otherwise, the graph will have the same number of connected components as before removing v^*, but G will contain one less vertex. This can be summarized as,*

$$\sum_{t_i \in T(v)} \binom{|t_i|}{2} \tag{3}$$

where $|t_i|$ is the number of vertices in tree t_i. This sum can be computed during the DFS-search when a vertex is found to be a cut vertex, and hence incurs only $\mathcal{O}(1)$ overhead per edge.

Observation 4. *If $v \in V$ is a cut vertex it will be identified upon recursing after visiting the entire subtree of each of its children, respectively. However, the order in which vertices are visited during the DFS-based search does not guarantee that all non-descendant vertices in the graph will be explored before reaching v. Hence, the number of vertices in the ancestor DFS-tree of v can be determined by computing $|V(v)| - s(v)$, where $V(v)$ indicates the set of vertices reachable from v before it is removed. $V(v)$ can be monitored at run-time.*

The above observations imply that v^* can be computed in linear time by augmenting a DFS for cut vertices to additionally calculate the impact of removing any vertex $v \in V$, which can then be used to identify v^*,

$$v^* = \arg\min_{v \in V} f(v) = \arg\min_{v \in V} \left(|V(v)| - s(v) + \sum_{t_i \in T(v)} \binom{|t_i|}{2} \right) \tag{4}$$

Since DFS has running time complexity of $O(n + m)$ and Eq. (4) can be executed in constant time per node during the search, the proposed greedy algorithm requires $O(K(n + m))$ complexity to remove K vertices from G.

Observation 5. *Let $Q \subseteq V \setminus \{v^*\}$ be the subset of vertices not reachable in graph G from vertex v^*. Then, it is not necessary to recompute the impact of removing any $w \in Q$ from $G \setminus \{v^*\}$ since v^* and each w belong to different network components. That is, only vertices $u \in V(v^*)$ must be re-examined.*

Observation 6. *Each connected component $\mathcal{C}_i$ can be identified by a root vertex associated with a DFS tree. The vertex whose removal will maximally decrease the objective function can be recorded during the DFS search and returned as this root, along with $V(v)$. This requires no significant computational or memory overhead.*

Observations 5 and 6 indicate that further practical improvements are possible. Specifically, a priority queue can be implemented to store the set of connected components C, which are represented and ordered by their respective root vertices. For each C_i, removing its root vertex will most significantly decrease the CNDP objective value versus any other vertex in the same component. After v^* is removed from the queue, the component that had contained it will be re-evaluated using the modified DFS-search and any newly resulting connected components will be added to the priority queue. Depending on queue implementation, maintaining proper queue ordering requires $\mathcal{O}(\log |C|)$. The per-iteration runtime significantly improves as the number of vertices in each connected component decreases. Effectively, the expected computation time will be $\approx K\left(\frac{(n+m)}{|C|} + \log |C|\right)$, although the worst case remains $\mathcal{O}(K(m + n))$.

Table 1. Benchmark networks and their properties. $|V|$ and $|E|$ are the number of vertices and edges, ρ is the global clustering coefficient, δ is the diameter and ξ is the degree assortativity.

Network	Type	$\lvert V\rvert$	$\lvert E\rvert$	ρ	δ	ξ
Conmat [15]	Collaboration	23,133	93,439	0.264	15	0.134
Ego [16]	Social	4,039	88,234	0.519	8	0.064
Flight [18]	Transportation	2,939	15,677	0.255	14	0.051
Powergrid [29]	Power grid	4,941	6,594	0.103	46	0.003
Relativity [15]	Collaboration	5,242	14,484	0.630	17	0.659
Oclinks [19]	Social	1,899	13,838	0.057	8	-0.188

3 Experimental Results

We evaluate the algorithm using six benchmark real-world problems to demonstrate the practical approximation quality and runtime. The benchmark networks and their properties are given in Table 1. All networks are simplified before use (no self-loops or multi-edges). Our experimental results compare Algorithm 1 to three centrality measures used in greedy sequential fashion such as in [25]. These are used only as a base-level comparison for the quality of the greedy approach. We consider node degree, PageRank [5] and Authority score [13] centrality attacks.

3.1 Summary of Results

To compare the quality of the greedy approach we vary K as 0.01, 0.05, 0.10, 0.15, 0.20 and 0.25 proportion of the network, respectively. The computer used for simulations was a 3.4 GHz Intel i7 processor with 16 GB RAM, running openSUSE Linux. We compare results to other methods of network attack (degree, PageRank and Kleinberg's Authority score) in a similar greedy sequential approach. These strategies have been recognized as potentially useful for network fragmentation when considering other robustness measures such as minimizing the largest network component [25]. Due to limited space a more rigorous experimental analysis was not possible, which would include an expanded set of real-world and benchmark network instances as well as other CNDP algorithms such as [2,22]. It should be noted that betweenness and closeness centrality, which are often also employed to test network vulnerability were too computationally inefficient to be considered for these networks.

Table 2 compares the objective value for $K = \{0.10n, 0.20n\}$ of the vertices in each network, respectively. The greedy approach outperforms the alternative strategies in all cases. Figure 2 plots the same experimental results over the entire range of K values for each network where the significance of the greedy solution quality is better highlighted. The proposed algorithm is especially destructive

Table 2. Comparison of the CNDP objective value after removing 10 % and 20 % of vertices from each network in Table 1.

Problem	K	GREEDY	Degree	PageRank	Authority
Comnat	2313	58,796,393	103,398,683	87,630,163	126,804,602
	4627	83,686	90,610	92,242	7,399,785
Ego	404	2,717,347	5,339,614	3,816,109	6,320,816
	808	1,848,740	2,070,535	2,886,709	3,438,031
Flight	294	322,527	484,331	467,962	1,014,305
	588	1,457	1,698	1,715	1,567
Powergrid	494	22,182	51,508	212,369	56,815
	988	3,639	4,580	14,744	3,771
Relativity	524	224,010	1,628,337	302,309	3,382,195
	1,048	4,089	4,896	9,023	6,390
Oclinks	190	637,936	785,662	758,328	835,297
	380	218,215	258,277	246,876	306,289

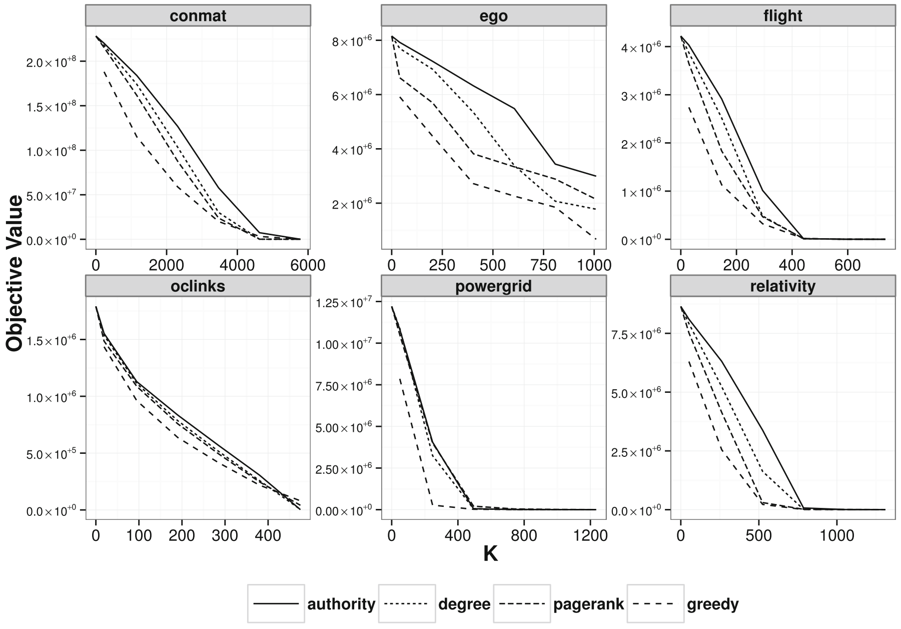

Fig. 2. Performance of each of the four strategies on the objective value. In all cases the greedy approach proposed in this paper yields the most desirable result.

Table 3. Comparing the run times (in milliseconds) of each approach. The proposed greedy approach is considered for both cases of using the queue-based strategy (fast greedy) or not (slow greedy).

Problem	Slow greedy	Fast greedy	Degree	PageRank	Authority
Conmat	1,732,841	21,229	379	16,810	26,027
Ego	7,036	2,129	92	1,480	3,750
Flight	10,199	207	25	144	231
Powergrid	29,896	252	41	288	3,470
Relativity	43,679	421	44	180	601
Oclinks	2,301	143	11	114	360

for $K < 0.15n$. All of the networks except the ego network exhibit a power-law degree distribution, which seems to be a major influence on the ability of fragmenting the networks for centrality-based approaches. The greedy algorithm significantly outperforms in these situations. Moreover, the global clustering coefficient, diameter or degree assortativity do not seem to have such an obvious impact as the degree distribution does.

Running time (in milliseconds) for these networks was also investigated and shown in Table 3. In order to highlight the benefit of the priority queue-based solution we sequentially remove vertices using both a naive greedy method that operates over the entire graph at each iteration (termed slow greedy) and the proposed fast greedy approach, which will only consider vertices in the component that contained the most recently removed vertex. The significant improvement of the priority queue is obvious. The greedy approach requires similar time to run as PageRank, although optimizing our implementation may further reduce or surpass this gap. As expected, sequentially removing vertices based on node degree is by far the fastest method.

4 Conclusion

In this paper we proposed an efficient greedy heuristic for identifying critical vertices in graphs whose removal leaves the residual graph with minimum pairwise connectivity. We particularly focused on larger graphs with many thousands of nodes, where finding solutions using current approaches typically requires a significant amount of time. We provided arguments for an upper-bound running time of $\mathcal{O}(K(n+m))$, although the practical performance is significantly improved using a priority-queue-based strategy for storing connected components. The resulting greedy algorithm is shown to yield better results than common centrality measures while being computationally competitive with degree-based greedy vertex removal.

The algorithm proposed in this paper was given without any proof of approximation quality. Future work will prove this bound. Moreover, experimentation on different network types and much larger sized, including runtime should also be conducted. Potential improvements in runtime may be attainable if within-connected component objective function evaluation was parallelized or a relationship between the nodes can be identified so that only an $\mathcal{O}(n)$ process is required to update the impact a vertex removal will have on its associated connected component. Extension to vertex and edge-weight variants of the CNDP is also possible, but not done in this paper.

References

1. Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. Discrete Appl. Math. **161**(16–17), 2349–2360 (2013)
2. Arulselvan, A., Commander, C.W., Elefteriadou, L., Pardalos, P.M.: Detecting critical nodes in sparse graphs. Comput. Oper. Res. **36**(7), 2193–2200 (2009)
3. Aspnes, J., Chang, K., Yampolskiy, A.: Inoculation strategies for victims of viruses and the sum-of-squares partition problem. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '05, pp. 43–52. Society for Industrial and Applied Mathematics (2005)
4. Boginski, V., Commander, C.: Identifying critical nodes in protein-protein interaction networks. In: Clustering Challenges in Biological, Networks, pp. 153–166 (2009)
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**, 107–117 (1998)
6. Chen, P., David, M., Kempe, D.: Better vaccination strategies for better people. In: Proceedings of the 11th ACM Conference on Electronic Commerce, pp. 179–188. ACM (2010)
7. Di Summa, M., Grosso, A., Locatelli, M.: Branch and cut algorithms for detecting critical nodes in undirected graphs. Comput. Optim. Appl. **53**, 649–680 (2012)
8. Di Summa, M., Grosso, A., Locatelli, M.: Complexity of the critical node problem over trees. Comput. Oper. Res. **38**(12), 1766–1774 (2011)
9. Dinh, T.N., Xuan, Y., Thai, M.T., Pardalos, P.M., Znati, T.: On new approaches of assessing network vulnerability: Hardness and approximation. IEEE/ACM Trans. Networking **20**(2), 609–619 (2012)
10. Dinh, T.N., Thai, M.T., Nguyen, H.T.: Bound and exact methods for assessing link vulnerability in complex networks. J. Comb. Optim. **28**(1), 3–24 (2014)
11. Joyce, K.E., Laurienti, P.J., Burdette, J.H., Hayasaka, S.: A new measure of centrality for brain networks. PLoS ONE **5**(8), e12200 (2010)
12. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence in a social network. In: Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
13. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)
14. Anil Kumar, V.S., Rajaraman, R., Sun, Z., Sundaram, R.: Existence theorems and approximation algorithms for generalized network security games. In: Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems, pp. 348–357 (2010)
15. Leskovec, J., Kleinberg, J.M., Faloutsos, C.: Graph evolution: densification and shrinking diameters. ACM Trans. Knowl. Dis. Data **1**(1), 2–41 (2007)

16. McAuley, J.J., Leskovec, J.: Learning to discover social circles in ego networks. In: NIPS, pp. 548–556 (2012)
17. Nguyen, D.T., Shen, Y., Thai, M.T.: Detecting critical nodes in interdependent power networks for vulnerability assessment. IEEE Trans. Smart Grid **4**(1), 151–159 (2013)
18. Opsahl, T.: Why anchorage is not (that) important: Binary ties and sample selection (2011). http://wp.me/poFcY-Vw
19. Opsahl, T., Panzarasa, P.: Clustering in weighted networks. Soc. Netw. **31**(2), 155–163 (2009)
20. Sporns, O.: Networks of the Brain. The MIT Press, Cambridge (2010)
21. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
22. 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)
23. Ventresca, M., Aleman, D.: Evaluation of strategies to mitigate contagion spread using social network characteristics. Soc. Netw. **35**(1), 75–88 (2013)
24. Ventresca, M., Aleman, D.: A derandomized approximation algorithm for the critical node detection problem. Comput. Oper. Res. **43**, 261–270 (2014)
25. Ventresca, M., Aleman, D.: Network robustness versus multi-strategy sequential attack. J. Complex Netw. (2014)
26. Ventresca, M., Aleman, D.: A randomized algorithm with local search for containment of pandemic disease spread. Comput. Oper. Res. **48**, 11–19 (2014)
27. Veremyev, A., Boginski, V., Pasiliao, E.L.: Exact identification of critical nodes in sparse networks via new compact formulations. Optim. Lett. **8**(4), 1245–1259 (2014)
28. 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)
29. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. Nature **393**, 400–442 (1998)