Ventresca and Aleman *Computational Social Networks* (2015) 2:6
DOI 10.1186/s40649-015-0010-y

Computational Social Networks
a SpringerOpen Journal

RESEARCH Open Access

Efficiently identifying critical nodes in large complex networks

Mario Ventresca[1*] and Dionne Aleman[2]

*Correspondence:
mventresca@purdue.edu
[1]School of Industrial Engineering, Purdue University, West Lafayette, USA
Full list of author information is available at the end of the article

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 for user-defined value k. Existing optimization algorithms are incapable of finding a good set R in graphs with many thousands or millions of vertices due to the associated computational cost. Hence, there exists a need for a time- and space-efficient approach for evaluating the impact of removing any $v \in V$ in the context of the CNDP. In this paper, we propose an algorithm based on a modified depth-first search that requires $O(k(|V| + |E|))$ time complexity. We employ the method within in a greedy algorithm for quickly identifying R. Our experimental results consider small- (≤ 250 nodes) and medium-sized ($\leq 25,000$ nodes) networks, where it is possible to compare to known optimal solutions or results obtained by other heuristics. Additionally, we show results using six real-world networks. The proposed algorithm can be easily extended to vertex- and edge-weighted variants of the CNDP.

Introduction

Detecting important or critical vertices in a graph/network has many important applications. Depending on the context, these critical vertices/nodes may be used to promote or mitigate a diffusive process that is acting upon the network. If promoting a spreading process, such as to spread market advertisements or public health warnings, the notion of 'critical' refers to the identification of individuals who are most likely to be influential spreaders and maximally permit information spread through the network. In such cases, the selected individuals may be targeted for demonstrations and promotions or invited to public events. For mitigation contexts, such as stopping the spread of a computer virus or for the construction of stable power delivery networks, the identified vertices are those whose removal from the graph will maximally limit diffusive spread.

Numerous definitions of what a critical node 'is' have been previously investigated, including junctions in cell-signaling or protein-protein networks [1], highly influential individuals [2], smart grid vulnerability [3], targeted vaccination for pandemic prevention [4,5], or keys to decipher brain functionality [6]. In some contexts, an accurate mathematical definition for a critical node, particularly for highly complex systems such as the brain [7], may not yet exist. It is important to note that both promotion and mitigation can often be defined in a mathematically similar manner. In this paper, we focus on the

context of mitigation; however, the results are typically applicable for problems where the goal is to maximally aid the diffusive process [8-11].

More specifically, in this paper, we consider the critical node detection problem (CNDP) as defined by [12]. 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 residual graph $G \setminus R$ 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, and $|\mathcal{C}_i|$ indicates the number of vertices in component $\mathcal{C}_i$. The optimal network is therefore one that is maximally fragmented and simultaneously minimizes the variance among the number of vertices in the connected components. That is, the residual network will contain a relatively large set of connected components, each containing a similar number of vertices. This problem has been shown to be $\mathcal{NP}$ hard [12,13]. Figure 1 highlights an example graph before and after removing all of its cut vertices (as an illustrative and simple strategy for detecting critical nodes) and the associated CNDP solution.

Related work

In general, graph partitioning has been an extremely active area of research for decades and we do not attempt a comprehensive review of those works here. Rather, in this section, we focus on the most related previous research to the CNDP and only highlight related works on graph partitioning for completeness.

The case where G is a tree structure has been examined and proven to be $\mathcal{NP}$ complete when considering non-unit edge costs [14]. A polynomial-time dynamic programming algorithm with worst-case complexity $\mathcal{O}\left(n^3k^2\right)$ for solving the problem with unit edge costs was also provided and applied to variants of the CNDP [15]. In [16], an integer linear programming model with a non-polynomial number of constraints is given and branch-and-cut algorithms were proposed. A reformulation of the CNDP that requires $\Theta\left(n^2\right)$ constraints was recently shown and optimal solutions for small networks were ascertained [17,18].

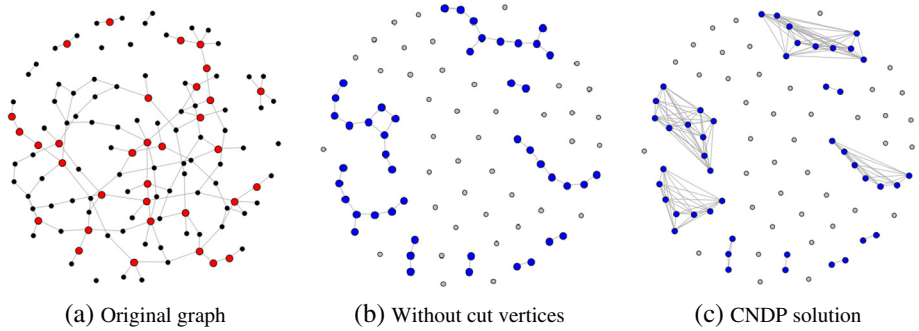

Figure 1 Visualizations of an initial input graph, residual graph, and corresponding CNDP solution. Visualizing an **(a)** initial input graph, the **(b)** residual graph after removing all cut vertices in the original graph (shown in red), and **(c)** the corresponding CNDP solution which highlights the fully connected subgraphs implied by Equation 1.

Heuristic solutions without provable approximation bounds have also been investigated, but computation time for very large networks remains an important issue in these methods as well. The CNDP work of [12] utilizes a solution to the maximum-independent set (MIS) problem as a starting point for a 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 [19] that permit solutions to significantly larger networks (having up to a few thousand vertices) to be solved within reasonable time and without significant resources. Randomized rounding-based algorithms have been also proposed in [20,21] 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 [13].

The CNDP is related to a variety of other graph partitioning problems in the literature; for instance, the minimum multi-cut problem, which aims to separate a set of source-sink pairs by removing a subset of minimum weighted edges. An $\mathcal{O}(\log n)$ approximation for general graphs is provided in [22]. Another well-studied graph partitioning problem is the k-cut problem. Given an undirected weighted graph, the goal is to find a minimum cost set of edges that separates the graph into at least k connected components. An $\mathcal{O}(2 - 2/k)$-approximation algorithm has been devised for this problem [23]. Classical multi-way cut, multi-cut, and k-cut problems that include a budget constraint to limit the number edges or vertices that can be cut have been studied in [24], where the authors also propose the problem of maximizing the number of connected components. In [25,26], an $\mathcal{O}(\sqrt{n})$-approximation algorithm is presented for the sparsest cut, edge expansion, balanced separator, and graph conductance problems, all of which are based on the notion of graph partitioning. These approaches also minimize the size of the interface between the resulting components. Their approach is based on semi-definite relaxation to these problems in concert with expander flows and has influenced much subsequent research.

Among the many problems that have been defined, some of the most similar include the following. The goal of the minimum contamination problem is to minimize the expected size of contamination by removing a set of edges of at most a given cardinality [27]. A variant of this problem is also proposed with the goal of minimizing the proportion of vertices in the largest resulting network, and 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 [28], a game-theoretic analysis is conducted that requires a solution to a generalization of the sum-of-squares partitioning problem [4]. Exact methods for link-based vulnerability assessment using edge disruptors have also recently been investigated [29,30].

The remainder of this paper is organized as follows. The proposed greedy algorithm and its motivation are provided in Section 'The proposed algorithm'. Section 'Experimental results' provides experimental results on small- and medium-sized benchmark networks, as well as six real-world networks, and a comparison to the greedy MIS-based algorithm of [12] is conducted. Conclusions are then presented in Section 'Conclusion'.

The proposed algorithm

In this section, we propose a greedy algorithm for the CNDP. Greedy algorithms are typically less computationally intensive than other strategies such as dynamic programming but usually sacrifice solution quality to attain this speed. One exception occurs in the case of maximizing a submodular function, where it was shown in [31] that unless $\mathcal{P} = \mathcal{NP}$, a

greedy algorithm will yield the optimal approximation. It is easy to verify that the CNDP is not a submodular function. However, the greedy approach is still an appealing framework in the CNDP context, especially for very large networks.

We first describe a linear-time algorithm to evaluate the impact of removing each $v \in V$ and then use this within a greedy algorithm for determining a solution to Equation 1. The speed of the greedy algorithm is further enhanced by a priority-queue-based implementation that yields significant practical performance increases over a naive implementation. Our algorithm has a running-time complexity of $\mathcal{O}(k(n+m))$ and is similar to the $\mathcal{O}\left(kn^2\right)$ approach used for the maximum cascading algorithm in [3]. Both algorithms are based on Tarjan [32].

For large networks with many thousands or millions of vertices (and edges), a computationally efficient approach to minimizing the CNDP objective is required. Selecting a critical subset $R \subset V$ from a single observation of the network may be easily deceived due to the influence of cut vertices, as indicated in Lemma 1 [12]. That is, selecting R in a sequential fashion may better 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 $\mathcal{C}_h = \mathcal{C}_\ell, \quad \forall h, \ell \in M_1$, then we obtain a better objective function value by deleting the set D_1 (where $\mathcal{C}_h$ is the number of vertices in connected component h).*

Thus, we propose the sequential greedy approach shown in Algorithm 1. At each iteration, the vertex whose removal will have the largest decrease on the objective function (Equation 1) is selected for removal and added to set R, where $f()$ computes the CNDP 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, which is too computationally intensive.

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 $|E| = 0$

Instead, we provide an $\mathcal{O}(k(n+m))$ algorithm based on a modified depth-first search (DFS). On a practical note, the iterative (versus recursive) algorithm implementation of DFS should be used because sufficiently large networks will quickly encounter stack overflow errors during the search. Performing a DFS on G will construct an equivalent graph

representation called a DFS-tree $DFS(v)$ rooted at arbitrary v. Figure 2 provides an example of the conversion between an original graph G to a DFS tree rooted at $v = 0$. Our subsequent solution is derived from observations of the DFS tree.

Observation 1. *Ignoring back edges, a leaf vertex $v \in V$ of a DFS tree has no children. Hence, the subtree rooted at v contains a single vertex whose deletion will not create any new connected components in the residual graph. That is, G and $G \setminus \{v\}$ contain the same number of 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 vertices descendant from v through these children can be recursively defined as*

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

So, upon removing a vertex $v^* \in V$, the residual graph will contain at least the same number of connected components as the previous graph. Now, let $T(v)$ denote the set of subtrees of v in a DFS tree, as represented by the immediate descendant of v. For instance, in Figure 2b, $T(3) = \{4\}$ and for $t_i \in T(v)$, $|t_i| = 5$. We can then make the following observations.

Observation 3. *Each internal vertex v of the DFS tree will either be a cut vertex or not. Removing $v^* = v$ will obviously result in an updated objective value, but if v^* is a cut vertex then the residual graph $G(V \setminus \{v\})$ will contain a nonempty $T(v^*)$ because at least one new connected component will be split from G. Ignoring back edges from v^*, the contribution of the children subtrees to the new objective value is computed as*

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

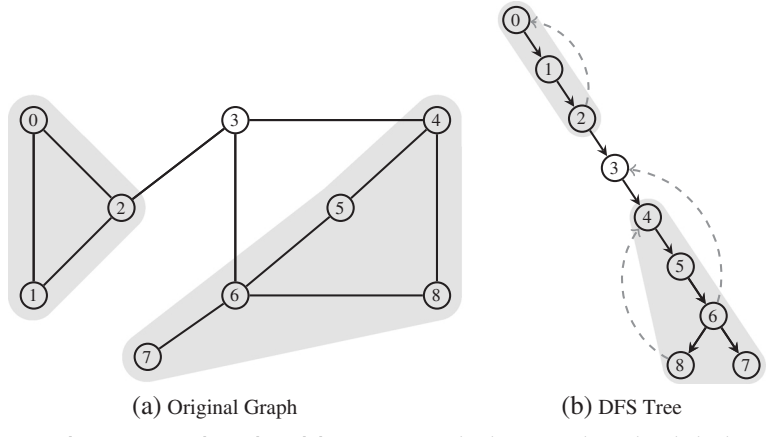

Figure 2 Equivalent (a) original graph and (b) DFS tree. Back edges are indicated as dashed arrows in DFS(0). The shaded areas correspond to resulting connected components if vertex $v = 3$ is removed from the graph. A traditional application of DFS is to detect cut vertices (i.e., articulation points), which forms the basis for our approach.

where $|t_i|$ is the number of vertices in DFS subtree $t_i \in T(v^*)$. As will be shown below, this sum can be straightforwardly computed for each vertex during the backtracking stage of DFS under the presumption that the current vertex being explored may be the next vertex removed.

Observation 4. *If v is a cut vertex, then it will be identified as being so after visiting the entire subtree of each of its children. However, the order in which vertices are visited during the DFS 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 must also be recorded. This is accomplished by computing the difference between the total number of vertices in the graph and those descendant vertices of v, i.e. $|V(v)| - s(v)$, where $V(v)$ indicates the set of vertices reachable from v. $V(v)$ is the size of the connected component to which v belongs and can be easily monitored at run time.*

The above four observations imply that v^* of Algorithm 1 can be computed in linear time by augmenting a DFS for identifying cut vertices to additionally calculate the impact of removing any vertex $v \in V$. That is,

$$v^* = \arg\min_{v \in V} f(v) = \arg\min_{v \in V} \left(f\left(\underbrace{|V(v)| - s(v) - 1}_{\text{ancestors}} \right) + \underbrace{\sum_{t_i \in T(v)} \binom{|t_i|}{2}}_{\text{descendants}} \right) \tag{4}$$

which is accomplished during the backtracking phase of DFS. Pseudocode for implementing the approach is given in Appendix A. Since DFS has running time complexity of $O(n + m)$ and Equation 4 can be executed in constant time per node during the search, then the proposed greedy algorithm requires $O(k(n+m))$ complexity to remove k vertices from G.

We make two further observations that will yield significant practical improvements by storing each connected component in a priority queue, indexed by the vertex whose removal in the component will most minimize Equation 1.

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 if v^* is deleted from G.*

Observation 6. *Each connected component $\mathcal{C}_i$ of graph G can be identified by a root vertex associated with a DFS tree. For each $\mathcal{C}_i$, there will exist a vertex v_i' whose removal maximally decreases the objective function value. Let v_i' be the root of the i^{th} DFS tree associated with $\mathcal{C}_i$. This requires no significant computational or memory overhead since upon deletion of v^*, the subgraph to which it belongs must be re-evaluated with respect to the objective function. The proposed algorithm in Appendix A will successfully determine v_i'.*

Observations 5 and 6 indicate that further practical improvements are possible. Specifically, a priority queue can be utilized to store the set of connected components C, which

are represented and ordered by their respective root vertices and their impact on the objective value if removed, respectively. 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 the component to which v^* belongs is removed from the queue, it will be re-evaluated using the modified DFS search and any newly resulting connected components will be added to the priority queue with an appropriate root node. Depending on the queue implementation, maintaining priority should require no more than $\mathcal{O}(\log |C|)$. The per-iteration run time will significantly improve 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 $\mathcal{O}(k(m+n))$. Algorithm 2 outlines the priority queue-based approach.

Algorithm 2 Fast vertex removal for CNDP.

find $q \in G$ using EVALUATE (Appendix A)
add q to priority queue
$R = \emptyset$
repeat
  remove v^* from priority queue
  $G = G \setminus \{v^*\}$
  $R = R \cup \{v^*\}$
  mark v^* as deleted
  while v^* has unvisited neighbors **do**
    get b the next unvisited neighbor v^*
    find $w \in G$, rooted at b using EVALUATE
    add w to the priority queue
  end while
until k vertices have been removed from G

Experimental results

We evaluate the proposed algorithm on three sets of data sets: (1) small networks where optimal solutions or bounds are known [20], (2) medium-sized benchmark networks [19], (3) six real-world networks. The real-world benchmark networks and their properties are given in Table 1. All networks are unweighted and simplified before use (no self-loops or multi-edges).

Table 1 Benchmark networks and their properties

Network	Type	$\|V\|$	$\|E\|$	ρ	δ	ξ
Conmat [33]	Collaboration	23,133	93,439	0.264	15	0.134
Ego [34]	Social	4,039	88,234	0.519	8	0.064
Flight [35]	Transportation	2,939	15,677	0.255	14	0.051
Powergrid [36]	Power grid	4,941	6,594	0.103	46	0.003
Relativity [33]	Collaboration	5,242	14,484	0.630	17	0.659
Oclinks [37]	Social	1,899	13,838	0.057	8	−0.188

$|V|$ 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.

We compare results to the most similar approach in literature [12], which is an $\mathcal{O}\left(n^2m\right)$ greedy algorithm based on MIS and local search. Unless stated otherwise, the MIS-based algorithm is run for k iterations (for similarity to the proposed algorithm) with one iteration during the local search phase. For medium-sized real-world networks, our experimental results also compare Algorithm 1 to three centrality measures used in greedy sequential fashion such as in [38]. These are used only as a base-level comparison for the quality of the greedy approach. We consider node degree, PageRank [39], and authority score [40] centrality attacks. The computer used for simulations was a 3.4 GHz Intel i7 processor with 16 GB RAM, running Linux Mint Debian Edition.

Small networks

Even though the purpose of the proposed algorithm is to provide a means of determining a set of critical nodes in very large networks, an analysis of its performance on small networks is useful to gauge approximation ability. The running time in all cases in negligible (less than 1 s). The MIS-based approach is also implemented for comparison and required up to 30 s to complete. There are four network types, and the number of nodes to remove is varied: $k = \{5, 10, 15, \ldots, 50\}$. Gurobi Solver 5.6 [41] is used to determine the upper and lower bound, within a 3,000-s time limit.

The results are summarized in Table 2 and Figure 3. The main difference between the two greedy algorithm results is that for highly connected networks, the MIS-based approach seems to be more useful; whereas the proposed algorithm is better suited for sparse networks.

Medium-sized networks

Table 3 presents results using 16 benchmark instances found in [19]. The table indicates the results of the PBIL algorithm [19], which is a population-based search, as well as sequential node removal using maximum node degree and highest PageRank as a heuristic objective, respectively. Of course, for these latter two experiments, the reported result indicates the CNDP objective of Equation 1. Values indicated in bold are the best results among those observed for each problem instance.

The benchmark results in Tables 2 and 3 reveal two insights. Firstly, sequential algorithms may perform poorly when compared to non-sequential algorithms in the instance of many potential solutions of equal quality. This is especially observed from the Watts-Strogatz network results. In these cases, the networks are highly connected and so there is unlikely to be many cut vertices. Moreover, as the networks are more sparse, the greedy approach becomes increasingly desirable. These two insights are founded in the same observation.

Observation 7 (The Problem of Ties). *Highly connected graphs with few cut vertices will admit numerous candidate solutions, each with similar objective value. Due to sequential-based approaches lacking ability to investigate sets of potential solutions, these algorithms are best suited for sparse graphs.*

The first consequence of The Problem of Ties concerns an explanation for the observed behavior, while the second leads to a conjecture.

Table 2 Results of the proposed greedy algorithm (SEQ) versus the MIS-based algorithm and upper (UB) and lower bound (LB) as determined by Gurobi 5.6 solver within 3,000 s

Graph	*k*	LB	UB	SEQ	MIS	Graph	*k*	LB	UB	SEQ	MIS
ER125	5	405	405	851	715	ER150	5	4,277	4,277	4,972	5,471
ER125	10	133	133	151	199	ER150	10	837	2,301	2,678	4,862
ER125	15	66	66	76	81	ER150	15	347	347	976	890
ER125	20	36	36	44	43	ER150	20	167	167	220	410
ER125	25	20	20	23	24	ER150	25	94	94	101	201
ER125	30	9	9	12	15	ER150	30	58	58	69	124
ER125	35	3	3	7	10	ER150	35	36	36	44	79
ER125	40	0	0	2	5	ER150	40	22	22	29	52
ER125	45	0	0	0	0	ER150	45	14	14	16	36
ER125	50	0	0	0	0	ER150	50	8	8	11	22
BA200	5	818	818	818	1,073	BA250	5	782	782	782	1,235
BA200	10	297	297	298	668	BA250	10	342	342	342	626
BA200	15	156	156	156	413	BA250	15	221	221	221	462
BA200	20	104	104	104	304	BA250	20	143	143	143	355
BA200	25	70	70	70	206	BA250	25	105	105	105	293
BA200	30	48	48	48	164	BA250	30	77	77	77	243
BA200	35	33	33	33	135	BA250	35	60	60	60	213
BA200	40	20	20	22	114	BA250	40	45	45	45	183
BA200	45	15	15	17	98	BA250	45	30	30	30	158
BA200	50	10	10	12	77	BA250	50	21	21	23	135
WS100	5	2,766	4,465	4,465	4,465	WS125	5	4,578	7,140	7,140	7,140
WS100	10	1,054	2,941	4,005	4,005	WS125	10	1,796	6,005	6,555	6,555
WS100	15	572	945	3,570	1,784	WS125	15	892	3,642	5,995	5,995
WS100	20	379	495	3,160	1,107	WS125	20	618	3,187	5,460	5,460
WS100	25	234	300	2,775	578	WS125	25	433	708	4,950	1,719
WS100	30	176	219	2,415	414	WS125	30	309	467	4,465	758
WS100	35	132	148	2,080	271	WS125	35	239	344	4,005	480
WS100	40	98	103	1,770	159	WS125	40	195	246	3,570	370
WS100	45	70	72	1,485	94	WS125	45	152	180	3,160	278
WS100	50	48	48	1,225	67	WS125	50	117	137	2,775	207
FF125	5	3,237	3,643	3,874	5,280	FF150	5	5,895	7,789	7,660	7,789
FF125	10	872	1,805	2,202	2,588	FF150	10	2,362	9,591	6,252	6,711
FF125	15	318	318	1,198	515	FF150	15	1,491	7,893	5,087	5,819
FF125	20	165	165	773	249	FF150	20	911	3,909	4,043	5,192
FF125	25	111	111	422	158	FF150	25	561	1,967	3,359	4,319
FF125	30	73	73	118	103	FF150	30	339	1,237	2,322	3,869
FF125	35	46	46	65	69	FF150	35	216	380	1,874	693
FF125	40	29	29	40	45	FF150	40	159	174	1,275	344
FF125	45	16	16	25	31	FF150	45	105	117	793	240
FF125	50	11	11	15	21	FF150	50	79	79	321	155

The networks are those presented in [20].

Theorem 1. *Assume G is sufficiently large and connected such that randomly removing k vertices is, with very high probability, unlikely to reveal* v^**. Then, the worst-case problem instance for the proposed greedy algorithm will result when* $G = (V, E)$ *contains no cut vertices initially but a single vertex* v^* *exists whose removal immediately uncovers a sequence of* $k - 1$ *residual graphs that each contain a cut vertex and whose union forms the optimal choice for set R.*

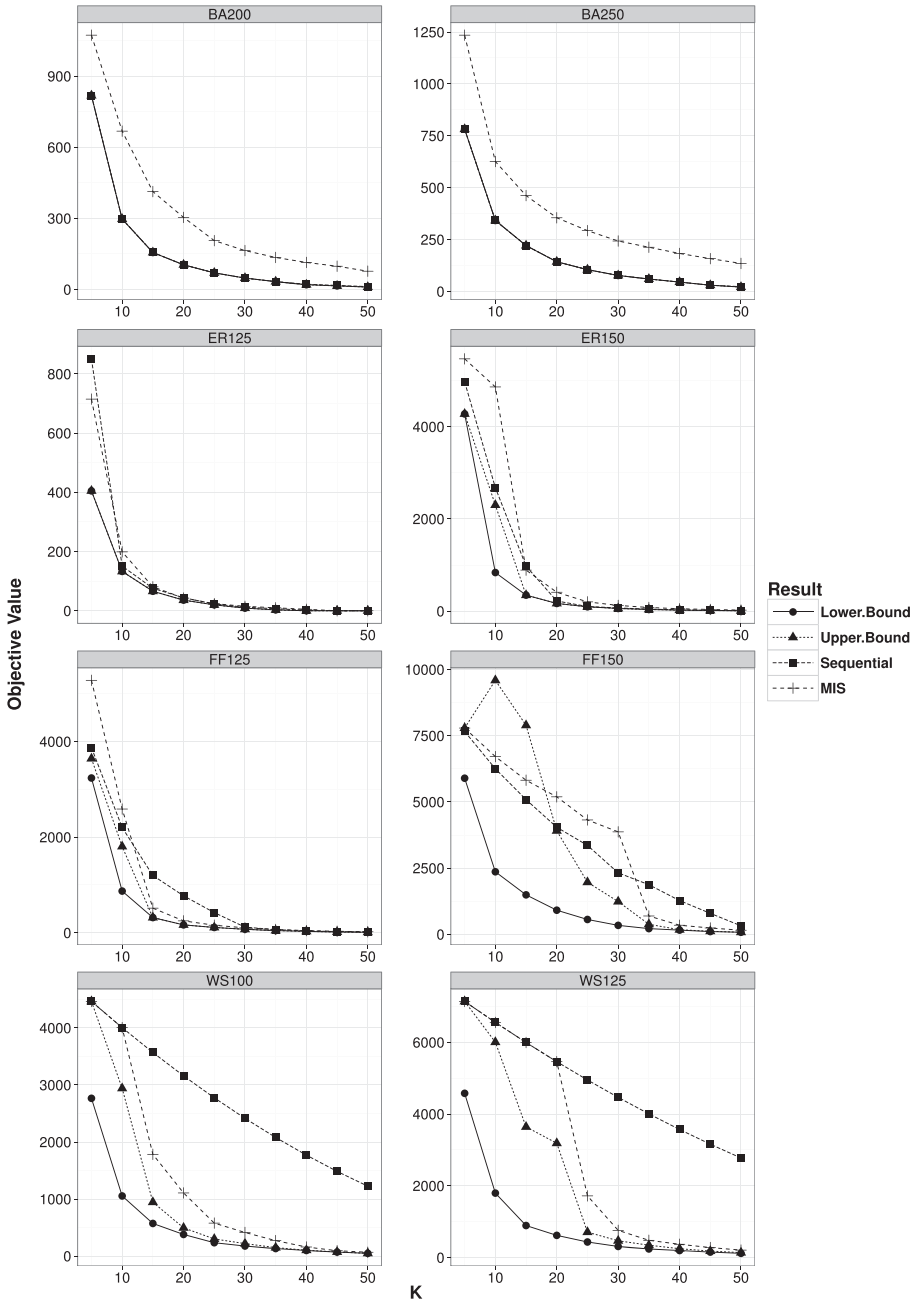

Figure 3 Plotting the results of Table 2.

Proof. Suppose there exists an optimal solution for the CNDP on G with cut-set R, $|R| = k$. Moreover, assume the residual graph $H = G \setminus R$ contains z connected components of equal size. Let a solution obtained by the greedy approach be composed of y components. Then, the objective function can be written as, for $z \leq y$,

$$z\binom{|\mathcal{C}_i|}{2} \leq \sum_{i=1}^{z}\binom{|\mathcal{C}_i|}{2} \leq \sum_{j=1}^{y}\binom{|\mathcal{C}_j|}{2} \tag{5}$$

Table 3 Summary benchmark results comparing PBIL [19], degree and PageRank-based sequential algorithms, and the proposed greedy approach

Problem	$\|V\|$	$\|E\|$	K	PBIL	Degree	PageRank	Greedy
ErdosRenyi	235	350	50	6,700	*1,086*	1,953	3,011
ErdosRenyi	466	700	80	44,255	*9,299*	25,892	28,994
ErdosRenyi	941	1,400	140	225,576	123,947	*113,752*	116,135
ErdosRenyi	2,344	3,500	200	2,009,132	1,851,950	1,708,603	*1,395,584*
BarabasiAlbert	500	499	50	892	202	236	*199*
BarabasiAlbert	1,000	999	75	3,057	622	689	*559*
BarabasiAlbert	2,500	2,499	100	28,044	4,258	4,808	*3,726*
BarabasiAlbert	5,000	4,999	150	146,753	13,038	13,971	*10,216*
WattsStrogatz	250	1,246	70	*13,768*	16,110	16,110	16,110
WattsStrogatz	500	1,496	125	*53,779*	70,125	62,149	69,751
WattsStrogatz	1,000	4,996	200	*308,596*	319,600	319,600	319,600
WattsStrogatz	1,500	4,498	265	*703,241*	761,995	700,474	761,995
ForestFire	250	514	50	886	547	403	*217*
ForestFire	500	828	110	1,904	902	490	*293*
ForestFire	1,000	1,817	150	9,594	7,796	2,609	*1,414*
ForestFire	2,000	3,413	200	12,569	27,451	11,419	*5,002*

Italic values indicate best solution per problem instance. Problem instances are those in [19].

If k vertices are removed in both the optimal and greedy solutions, then if greedy never encounters v^* or reveals any cut vertices,

$$z\binom{|\mathcal{C}_i|}{2} = z\binom{\frac{|V|-K}{z}}{2} \quad \text{and} \quad \sum_{j=1}^{y}\binom{|\mathcal{C}_j|}{2} = \binom{|V|-K}{2} \tag{6}$$

The exact value of z will depend on the number of new connected components created as each vertex is removed (after the initial v^*). Of course, no better optimal solution can be constructed in this circumstance. Moreover, the greedy algorithm will attain the worst possible objective value since no connected components will be created as a result of removing k vertices. □

Equation 6 also reveals insight into why highly connected networks are more difficult. That is, why the problem of ties is confounding. The difference between these two values will be smallest if many cut vertices exist, with few ties between solutions. Hence, we conjecture that reducing the Problem of Ties will result in higher quality solutions.

Real-world networks

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 each network, respectively. As with medium-sized networks, we compare results to 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 [38]. It should be noted that betweenness and closeness centrality, which are often also employed to test network vulnerability, are too computationally inefficient to be considered for these networks. We also compare the results to the MIS-based greedy heuristic. However, due to

excessive computation time, we only report the result after one iteration of the MIS-based algorithm.

Table 4 compares the objective value for $k = \{0.10n, 0.20n\}$ of the vertices in each network, respectively. The greedy approach outperforms the centrality-based strategies in all cases. The MIS-based approach is competitive with SEQ, but the computation time requirements limited its ability to discover a solution for the conmat network (see Table 5). Figure 4 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 versus the centrality-based approaches. The proposed algorithm is especially destructive 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 5. 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 (SEQ), 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. The MIS-based approach is significantly more time consuming, which is expected based on its $\mathcal{O}\left(n^2m\right)$ running time behavior.

Conclusion

In this paper, we proposed an efficient greedy heuristic for identifying critical vertices in networks whose removal leaves the residual network with minimum pairwise

Table 4 Comparison of the CNDP objective value after removing 10% and 20% of vertices from each network in Table 1

Problem	*K*	SEQ	Degree	PageRank	Authority	MIS
Comnat	2313	58,796,393	103,398,683	87,630,163	126,804,602	NA
	4627	83,686	90,610	92,242	7,399,785	NA
Ego	404	2,717,347	5,339,614	3,816,109	6,320,816	2,192,636
	808	1,848,740	2,070,535	2,886,709	3,438,031	903,441
Flight	294	322,527	484,331	467,962	1,014,305	77,777
	588	1,457	1,698	1,715	1,567	2,626
Powergrid	494	22,182	51,508	212,369	56,815	25,253
	988	3,639	4,580	14,744	3,771	5,378
Relativity	524	224,010	1,628,337	302,309	3,382,195	23,620
	1,048	4,089	4,896	9,023	6,390	6,163
Oclinks	190	637,936	785,662	758,328	835,297	746,085
	380	218,215	258,277	246,876	306,289	402,824

SEQ-based results are those obtained by the proposed algorithm. The MIS-based approach was unable to arrive at solutions within 40 h for the conmat networks.

Table 5 Comparing the run times (in milliseconds) of each approach

Problem	Slow greedy	SEQ	Degree	PageRank	Authority	MIS
Conmat	1,732,841	21,229	379	16,810	26,027	126,101,000
Ego	7,036	2,129	92	1,480	3,750	111,931,000
Flight	10,199	207	25	144	231	3,273,000
Powergrid	29,896	252	41	288	3,470	22,292,000
Relativity	43,679	421	44	180	601	43,016,000
Oclinks	2,301	143	11	114	360	1,734,000

The proposed greedy approach is considered for both cases of using the queue-based strategy (fast greedy) or not (slow greedy).

connectivity. 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. We utilized both benchmark and larger networks with many thousands of nodes, where finding solutions using current approaches typically requires a significant amount of time. The resulting greedy algorithm is shown to yield better results than common centrality measures for large graphs while being computationally competitive with degree-based greedy vertex removal. The results on benchmark graphs led to the abstract construction of a worst-case input graph, which was a consequence of identifying The Problem of Ties. Interesting future work may aim to reduce the impact of this issue by incorporating additional information that is highly correlated with the objective in order to better identify potentially interesting candidate nodes for removal.

The algorithm proposed in this paper was given without any proof of approximation quality, only indicating the extreme circumstances of problem instance. Future work will prove this bound. Moreover, experimentation on different network types and much larger

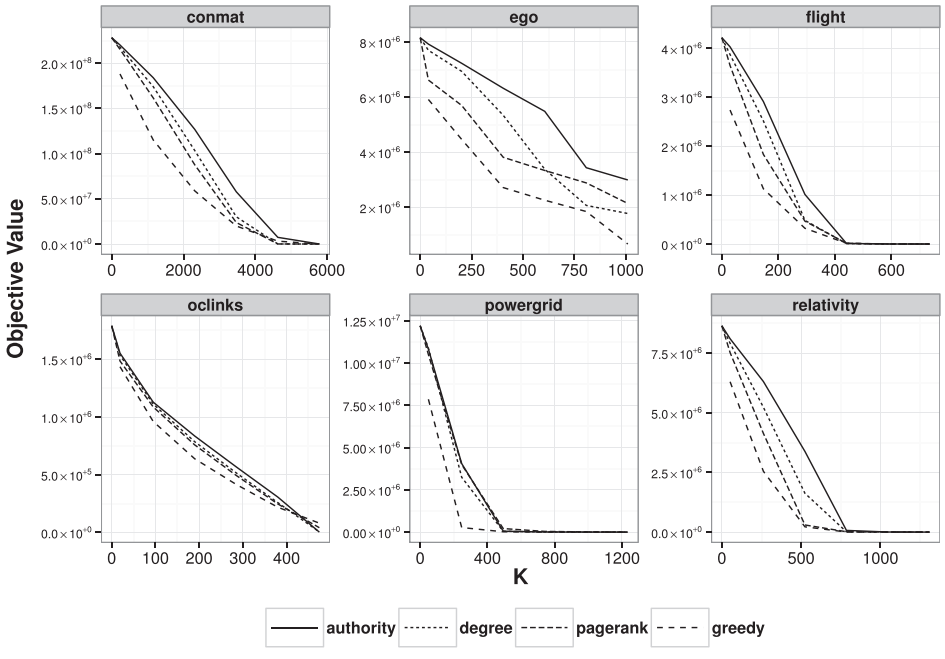

Figure 4 Performance of the SEQ and three centrality-based strategies on the objective value. In all cases, the greedy approach proposed in this paper yields the most desirable result.

sized networks, including run time, should also be conducted. Potential improvements in run time 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 of a vertex removal. Extensions to vertex- and edge-weighted variants of the CNDP are also possible.

Appendix

A Evaluating objective function in $\mathcal{O}(|V| + |E|)$

Algorithm 3 The pseudocode for EVALUATE.

```
Require: Stack S, DFN, LOW, COUNTED, PARENT, CUT_SIZE, ST_SIZE, IMPACT
num = 1, push(root,S)
repeat
  Vertex v = peek(S)
  y = next unvisited neighbour of v
  if y exists then
    mark y as visited, push(y,S)
    DFN[y] = num; LOW[y] = DFN[y]; PARENT[y] = x;
    ST_SIZE[y] = 1; IMPACT[y] = 0
    num = num + 1
  else
    pop(S) and add v to list of removed vertices
    for all neighbors w of v do
      if DFN[w] < DFN[v] and PARENT[v] ≠ y then
        LOW[v] = min(LOW[v],DFN[v])
      else
        LOW[v] = min(LOW[v], LOW[w])
        if !COUNTED[w] and (PARENT[v] != w OR v is the root) then
          COUNTED[w] = true
          ST_SIZE[v] = ST_SIZE[v] + ST_SIZE[w]
        end if
        if LOW[w] ≥ DFN[v] and v is not the root then
          mark v as an articulation point
          CUT_SIZE[v] = CUT_SIZE[v] + ST_SIZE[w]
          IMPACT[v] = IMPACT[v] + f(ST_SIZE[w])
        end if
      end if
      if v is the root and it has more than one child then
        mark root as an articulation point
      end if
    end for
  end if
until done
for each visited vertex v do
  if v is an articulation point then
    IMPACT[v] = IMPACT[v] + f(num - CUT SIZE[v])
  else
    IMPACT[v] = IMPACT[v] + f (num - 1)
  end if
  maintain the vertex v* with minimum IMPACT value
end for
return v*
```

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
MV devised and analyzed experiments, and drafted the initial manuscript. DA devised experiments and edited the manuscript. Both authors read and approved the final manuscript.

Author details
[1] School of Industrial Engineering, Purdue University, West Lafayette, USA. [2] Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada.

Received: 30 October 2014 Accepted: 19 January 2015
Published online: 28 March 2015

References

1. Boginski, V, Commander, C: Identifying critical nodes in protein-protein interaction networks. In: Clustering Challenges in Biological Networks, pp. 153–166. Elsevier, Amsterdam, Netherlands, (2009)
2. 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. ACM, New York, NY, US, (2003)
3. Nguyen, DT, Shen, Y, Thai, MT: Detecting critical nodes in interdependent power networks for vulnerability assessment. IEEE Trans. Smart Grid **4**(1), 151–159 (2013)
4. 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, Society for Industrial and Applied Mathematics, pp. 43–52. ACM, New York, NY, USA, (2005)
5. Ventresca, M, Aleman, D: Evaluation of strategies to mitigate contagion spread using social network characteristics. Soc. Netw. **35**(1), 75–88 (2013)
6. Joyce, KE, Laurienti, PJ, Burdette, JH, Hayasaka, S: A new measure of centrality for brain networks. PLoS ONE. **5**(8), e12200 (2010)
7. Sporns, O: Networks of the Brain. The MIT Press, Cambridge, MA, USA (2010)
8. Borge-Holthoefer J, Moreno, Y: Absence of influential spreaders in rumor dynamics. Phys. Rev. E. **85**, 026116 (2012)
9. Kempe, D, Kleinberg, J, Tardos, E: Influential nodes in a diffusion model for social networks. In: Proceedings of the 32nd international conference on Automata, Languages and Programming, ICALP'05, pp. 1127–1138. Springer, Berlin Heidelberg, (2005)
10. Kitsak, M, Gallos, LK, Havlin, S, Liljeros, F, Muchnik, L, Stanley, HE, Makse, HA: Identification of influential spreaders in complex networks. Nat. Phys. **6**(11), 888–893 (2010)
11. Richardson, M, Domingos, P: Mining knowledge-sharing sites for viral marketing. In: Proceedings 8th International Conference on Knowledge Discovery and Data Mining, pp. 61–70. ACM, New York, NY, USA, (2002)
12. Arulselvan, A, Commander, CW, Elefteriadou, L, Pardalos, PM: Detecting critical nodes in sparse graphs. Comput. Oper. Res. **36**(7), 2193–2200 (2009)
13. Dinh, TN, Xuan, Y, Thai, MT, Pardalos, PM, Znati, T: On new approaches of assessing network vulnerability: hardness and approximation. IEEE/ACM Trans. Netw. **20**(2), 609–619 (2012)
14. Di Summa, M, Grosso, A, Locatelli, M: Complexity of the critical node problem over trees. Comput. Oper. Res. **38**(12), 1766–1774 (2011)
15. Addis, B, Di Summa, M, Grosso, A: 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)
16. Di Summa, M, Grosso, A, Locatelli, M: Branch and cut algorithms for detecting critical nodes in undirected graphs. Computational Optimization and Applications. **53**(3), 649–680 (2012)
17. Veremyev, A, Boginski, V, Pasiliao, EL: Exact identification of critical nodes in sparse networks via new compact formulations. Optimization Lett. **8**(4), 1245–1259 (2014)
18. Veremyev, A, Prokopyev, OA, Pasiliao, EL: An integer programming framework for critical elements detection in graphs. J. Comb. Optimization. **28**(1), 233–273 (2014)
19. 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)
20. Ventresca, M, Aleman, D: A derandomized approximation algorithm for the critical node detection problem. Comput. Oper. Res. **43**, 261–270 (2014)
21. Ventresca, M, Aleman, D: A randomized algorithm with local search for containment of pandemic disease spread. Comput. Oper. Res. **48**, 11–19 (2014)
22. Garg, N, Vazirani, V, Yannakakis, M: Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica. **18**, 3–20 (1997)
23. Saran, H, Vazirani, VV: Finding k-cuts within twice the optimal. SIAM J. Comput. **24**, 101–108 (1995)
24. Engelberg, R, Konemann, J, Leonardi, S, Naor, J: Cut problems in graphs with a budget constraint. In: Proceedings of the 7th Latin American Theoretical Informatics Symposium. Elsevier, Amsterdam, Netherlands, (2006)
25. Arora, S, Rao, S, Vazirani, U: Expander flows, geometric embeddings and graph partitioning. In: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pp. 222–231. ACM, New York, NY, USA, (2004)
26. Arora, S, Rao, S, Vazirani, U: Expander flows, geometric embeddings and graph partitioning. J. ACM **56**(2), 1–37 (2009)
27. Anil Kumar, VS, 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. IEEE, New Jersey, USA, (2010)
28. 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, New York, NY, USA, (2010)

29. Dinh, TN, Thai, MT, Nguyen, HT: Bound and exact methods for assessing link vulnerability in complex networks. J. Comb. Optimization. **28**(1), 3–24 (2014)
30. Shen, Y, Nguyen, NP, Xuan, Y, Thai, MT: On the Discovery of Critical Links and Nodes for Assessing Network Vulnerability. IEEE/ACM Transactions on Networking. **21**(3), 963-973 (2013)
31. Nemhauser, GL, Wolsey, LA, Fisher, ML: An analysis of approximations for maximizing submodular set functions-i. Math. Program. **14**(1), 265–294 (1978)
32. Tarjan, R: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
33. Leskovec, J, Kleinberg, JM, Faloutsos, C: Graph evolution: Densification and shrinking diameters. ACM Trans. Knowl. Discov. Data. **1**(1) (2007). http://doi.acm.org/10.1145/1217299.1217301
34. McAuley, JJ, Leskovec, J: Learning to discover social circles in ego networks. In: NIPS, pp. 548–556. ACM, New York, NY, USA, (2012)
35. Opsahl, T: Why anchorage is not (that) important: Binary ties and sample selection. (2011) http://wp.me/poFcY-Vw
36. Watts, DJ, Strogatz, SH: Collective dynamics of 'small-world' networks. Nature. **393**, 400–442 (1998)
37. Opsahl, T, Panzarasa, P: Clustering in weighted networks. Soc. Netw. **31**(2), 155–163 (2009)
38. Ventresca, M, Aleman, D: Network robustness versus multi-strategy sequential attack. J. Complex Netw. **3**(1), 126–146 (2015)
39. Brin, S, Page, L: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**, 107–117 (1998)
40. Kleinberg, JM: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)
41. Gurobi Optimization, Inc.: Gurobi Optimizer Reference Manual (2015). http://www.gurobi.com