Computers & Operations Research 36 (2009) 2193 – 2200

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

Detecting critical nodes in sparse graphs

Ashwin Arulselvan[a], Clayton W. Commander[b,*], Lily Elefteriadou[c], Panos M. Pardalos[a]

[a]Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
[b]Air Force Research Laboratory, Munitions Directorate, and Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
[c]Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL, USA

ARTICLE INFO

Available online 29 August 2008

Keywords:
Critical node detection
Heuristics
Integer linear programming
NP-complete
Combinatorial optimization

ABSTRACT

Identifying critical nodes in a graph is important to understand the structural characteristics and the connectivity properties of the network. In this paper, we focus on detecting critical nodes, or nodes whose deletion results in the minimum pair-wise connectivity among the remaining nodes. This problem, known as the CRITICAL NODE PROBLEM has applications in several fields including biomedicine, telecommunications, and military strategic planning. We show that the recognition version of the problem is $\mathcal{NP}$-complete and derive a mathematical formulation based on integer linear programming. In addition, we propose a heuristic for the problem which exploits the combinatorial structure of the graph. The heuristic is then enhanced by the application of a local improvement method. A computational study is presented in which we apply the integer programming formulation and the heuristic to real and randomly generated data sets. For all instances tested, the heuristic is able to efficiently provide optimal solutions in a fraction of the time required by a commercial software package.

Published by Elsevier Ltd.

1. Introduction

Given a graph and an integer k, the objective of the CRITICAL NODE PROBLEM (CNP) is to find a set of k nodes in the graph whose deletion results in the maximum network fragmentation. By this we mean, minimize the pair-wise connectivity between the nodes in the k-vertex deleted subgraph. Similar studies appearing in the social networks literature include those by Bavelas [1] and Freeman [2] which emphasize node centrality and prestige, both of which are usually functions of node degree. However, they lack applications to problems which emphasize network fragmentation and connectivity.

The CNP has several applications in the field of social network analysis. Social networks have attracted a significant amount of attention in recent years. The study of these graphs is important to better understand several properties which are most common in network depictions of social interactions including cohesion, transitivity, and centrality of specific actors of the graph [3]. The study of the various properties of social networks such as diameter, radiality, and connectivity are responsible for social contagion and provide scope for containment of an epidemic outbreak. These properties also help in designing strategies for communication breakdowns in human and telecommunication networks [4].

The CNP finds applications in network immunization [5,6] where mass vaccination is an expensive process and only a specific number of people, modeled as nodes of a graph, can be vaccinated. The immunized nodes cannot propagate the virus and the goal is to identify the individuals to be vaccinated in order to reduce the overall transmissibility of the virus. There are several vaccination strategies in the literature (see e.g., [5,6]) offering control of epidemic outbreaks; however, none of the proposed are optimal strategies. The vaccination strategies suggest emphasizing the *centrality* of nodes as a major factor rather than *critical* nodes whose deletion will maximize the disconnectivity of the graph. Deletion of central nodes may not guarantee a fragmentation of the network or even disconnectivity, in which case disease transmission cannot be prevented. Because social networks model the patterns of humans, they vary greatly over time. The relationships between people, represented by edges in the social network, are transient and there is a constant rewiring between the nodes as new relationships are established. The proposed critical node technique minimizes the transmission of the disease over an instance of the dynamic network.

Furthermore, the CNP can be applied to the study of covert terrorist networks, where a certain number of individuals have to be identified whose deletion will result in the maximum breakdown of communication between individuals in the network [7]. Likewise in order to stop the spreading of a virus over a telecommunication network, one can identify the critical nodes of the graph and take them

* Corresponding author.
E-mail addresses: ashwin@ufl.edu (A. Arulselvan), clayton.commander@eglin.af.mil, clayton.commander@gmail.com (C.W. Commander), elefter@ce.ufl.edu (L. Elefteriadou), pardalos@ufl.edu (P.M. Pardalos).

0305-0548/$ - see front matter Published by Elsevier Ltd.
doi:10.1016/j.cor.2008.08.016

offline. Similarly, if one's ultimate goal is to prevent communication on a wired telecommunication network, an efficient way of doing so would be to jam the critical nodes. This has been studied in the context of wireless networks by Commander et al. in [8].

Before proceeding, we mention one final area in which the CNP finds several applications, and that is in the field of transportation engineering [9]. Two particular examples are as follows. In general, for transportation networks, it is important to identify critical nodes in order to ensure they operate reliably for transporting people and goods throughout the network. Further, in planning for emergency evacuations, identifying the critical nodes of the transportation network is crucial. The reason is two-fold. First, knowledge of the critical nodes will help in planning the allocation of resources during the evacuation. Secondly, in the aftermath of a disaster they will help in re-establishing critical traffic routes.

Borgatti [10] has studied a similar problem, focusing on node detection resulting in maximum network disconnectivity. Other studies in the area of node detection such as centrality [1,2] focus on the prominence and reachability to and from the central nodes. However, little emphasis is placed on the importance of their role in the network connectivity and diameter. Perhaps one reason for this is that all of the aforementioned references relied on simulation to conduct their studies. Although the simulations have been successful, a mathematical formulation is essential for providing insight and helping to reveal some of the fundamental properties of the problem [11]. In the next section, we present a mathematical model based on integer linear programming which provides optimal solutions for the CNP.

We organize this paper by first formally defining the problem and discussing its computational complexity. Next, we provide an integer programming (IP) formulation for the corresponding optimization problem. In Section 3 we introduce a heuristic to quickly provide solutions for instances of the problem. We present a computational study in Section 4, in which we compare the performance of the heuristic against the optimal solutions which were computed using a commercial software package. Some concluding remarks are given in Section 5.

2. Problem definition

The formal definition of the problem is given by
CRITICAL NODE PROBLEM (CNP).
INPUT: An undirected graph $G=(V,E)$ and an integer k.
OUTPUT: $A=\operatorname{argmin}\sum_{i,j\in(V\setminus A)}u_{ij}(G(V\setminus A)) : |A|\leqslant k$, where

$$u_{ij} := \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in the same component of } G(V\setminus A),\\ 0 & \text{otherwise.}\end{cases}$$

The objective is to find a subset $A\subseteq V$ of nodes such that $|A|\leqslant k$, whose deletion minimizes the pair-wise connectivity among the nodes in the induced subgraph $G(V\setminus A)$.

This problem is similar to the MINIMUM k-VERTEX SHARING [12], where the objective is to minimize the number of nodes deleted to achieve a k-way partition. Here we are considering the complementary problem, where we know the number of vertices to be deleted and we try to maximize the number of components formed and implicitly limit the sizes of the components. Borgatti [10] has given a comprehensive illustration to facilitate the understanding of the objective function and its non-triviality.

2.1. Computational complexity

We now prove that the recognition version of the CNP is $\mathcal{NP}$-complete. Consider the recognition version of the CNP:
K-CRITICAL NODE PROBLEM (K-CNP)
INPUT: An undirected graph $G=(V,E)$ and an integer k.
QUESTION: Is there a set M, where M is the set of all maximal connected components of G obtained by deleting k nodes or less, such that $\sum_{h\in M}(\sigma_h(\sigma_h-1))/2\leqslant K$, where σ_h is the cardinality of component h, for each $h\in M$?

In order to prove that the K-CNP is $\mathcal{NP}$-complete, we make use of the following lemmata. In particular, we prove that optimizing the objective function not only maximizes the pair-wise disconnectivity among the nodes, but also minimizes the variance in the cardinalities of the components. Particularly, in Lemma 1 we show that for any two solutions resulting in the same number of components, if the cardinalities of the components are equal in one solution, and not equal in the other, then the objective value of the latter will always be worse than that of the former.

Lemma 1. *Let M be a partition of $G=(V,E)$ in to L components obtained by deleting a set D of nodes, where $|D|=k$. Then the objective function $\sum_{h\in M}(\sigma_h(\sigma_h-1))/2\geqslant((|V|-k)((|V|-k)/L-1))/2$, with equality holding if and only if $\sigma_h=\sigma_l, \forall h,l\in M$, where σ_h is the size of hth component of M.*

Proof. *Case* 1: $\boldsymbol{\sigma_h\neq\sigma_l}$, $\forall \mathbf{h},\mathbf{l}\in\mathbf{M}$.

Note that $\sum_{h\in M}\sigma_h=|V|-k$. Then given a solution for an instance of the CNP, we have that

$$\frac{1}{L}\sum_{h\in M}\left(\sigma_h-\frac{1}{L}\sum_{h\in M}\sigma_h\right)^2=\frac{1}{L}\sum_{h\in M}\sigma_h^2-\left(\frac{1}{L}\sum_{h\in M}\sigma_h\right)^2 \tag{1}$$
$$\geqslant 0 \tag{2}$$
$$=\left(\frac{|V|-k}{L}\right)^2-\left(\frac{|V|-k}{L}\right)^2. \tag{3}$$

This implies that

$$\frac{1}{L}\sum_{h\in M}\sigma_h^2\geqslant\left(\frac{|V|-k}{L}\right)^2. \tag{4}$$

Therefore,

$$\frac{1}{L}\sum_{h\in M}\sigma_h^2-\frac{1}{L}\sum_{h\in M}\sigma_h\geqslant\left(\frac{|V|-k}{L}\right)^2-\frac{1}{L}\sum_{h\in M}\sigma_h \tag{5}$$
$$=\left(\frac{|V|-k}{L}\right)^2-\frac{|V|-k}{L}. \tag{6}$$

Thus, we have that

$$\sum_{h\in M}\frac{\sigma_h(\sigma_h-1)}{2}\geqslant\frac{(|V|-k)\left(\frac{|V|-k}{L}-1\right)}{2}. \tag{7}$$

Case 2: $\boldsymbol{\sigma_h=\sigma_l}$, $\forall \mathbf{h},\mathbf{l}\in\mathbf{M}$.

In this case, each component of M will be of size $(|V|-k)/L$, which is obviously the average size of a component of M. Thus

$$\sum_{h\in M}\frac{\sigma_h(\sigma_h-1)}{2}=\frac{(|V|-k)\left(\frac{|V|-k}{L}-1\right)}{2}. \tag{8}$$

Conversely, if (8) holds, but each component of M is not the same size, it follows that not all components will be of average size and hence

$$\frac{1}{L}\sum_{h\in M}\sigma_h^2-\left(\frac{1}{L}\sum_{h\in M}\sigma_h\right)^2>0 \tag{9}$$
$$=\left(\frac{|V|-k}{L}\right)^2-\left(\frac{|V|-k}{L}\right)^2. \tag{10}$$

Similar to the above result, we see that

$$\frac{1}{L}\sum_{h\in M}\sigma_h^2 > \left(\frac{|V|-k}{L}\right)^2. \tag{11}$$

Thus,

$$\frac{1}{L}\sum_{h\in M}\sigma_h^2 - \frac{1}{L}\sum_{h\in M}\sigma_h > \left(\frac{|V|-k}{L}\right)^2 - \frac{|V|-k}{L} \tag{12}$$

$$\Rightarrow \sum_{h\in M}\frac{\sigma_h(\sigma_h-1)}{2} > \frac{(|V|-k)\left(\frac{|V|-k}{L}-1\right)}{2}. \tag{13}$$

This is a contradiction and we have the proof. □

The following lemma provides a similar result as above. However in this case, the number of components induced by each solution are not assumed to be equal.

Lemma 2. *Let M_1 and M_2 be two sets of partitions obtained by deleting D_1 and D_2 sets of nodes, respectively, from graph $G=(V,E)$, where $|D_1|=|D_2|=k$. Let L_1 and L_2 be the number components in M_1 and M_2, respectively, and $L_1 \geqslant L_2$. If $\sigma_h=\sigma_l, \forall h,l \in M_1$, then we obtain a better objective function value by deleting the set D_1.*

Proof. Let $f(M_1)$ and $f(M_2)$ be the objective function values obtained by deleting D_1 and D_2, respectively. Let us assume that $f(M_1)>f(M_2)$. Let $u=(|V|-k)/L_2$. From Lemma 1, we have that $(L_2(u)(u-1))/2 \leqslant f(M_2)$. Then we have

$$\frac{L_2(u)(u-1)}{2} \leqslant f(M_2) < f(M_1) = \frac{L_1\left(\frac{|V|-k}{L_1}\right)\left(\frac{|V|-k}{L_1}-1\right)}{2}. \tag{14}$$

Examining (14) carefully reveals the necessary contradiction. Notice that (14) states

$$\frac{L_2\left(\frac{|V|-k}{L_2}\right)\left(\frac{|V|-k}{L_2}-1\right)}{2} < \frac{L_1\left(\frac{|V|-k}{L_1}\right)\left(\frac{|V|-k}{L_1}-1\right)}{2} \tag{15}$$

$$\Rightarrow \frac{(|V|-k)(|V|-k-L_2)}{2} < \frac{(|V|-k)(|V|-k-L_1)}{2} \tag{16}$$

$$\Rightarrow L_2 > L_1. \tag{17}$$

This contradicts the hypothesis that $L_1 \geqslant L_2$, and we have the result. □

We can now prove the following theorem regarding the complexity of the CNP.

Theorem 1. *The K-CNP is $\mathcal{NP}$-complete.*

Proof. To show this, we must prove that (1) K-CNP $\in \mathcal{NP}$; (2) Some $\mathcal{NP}$-complete problem reduces to K-CNP in polynomial time.

(1) K-CNP $\in \mathcal{NP}$ since given any graph $G=(V,E)$, and deleting any set of at most k nodes, we can determine the objective value in $\mathcal{O}(|E|)$ time using a depth-first search [13].
(2) To complete the proof, we show a reduction from the INDEPENDENT SET PROBLEM (ISP) [14], which is well known to be $\mathcal{NP}$-complete [15]. Given a graph $G=(V,E)$, the ISP seeks to determine if G contains an independent set of size k. This is equivalent to determining if there exists an empty subgraph of G of size k by deleting $|V|-k$ nodes and their adjacent edges. Let $\bar{G}=(\bar{V},\bar{E})$ be the graph obtained by replacing each node $u \in G$ by a T-clique with one of the clique nodes coinciding with u. Note if $T=1$, then we have the original graph G. Consider the K-CNP on $\bar{G}$ which asks if there exists a partition M of $\bar{G}$ obtained by deleting $|V|-k$ nodes such that

$$\sum_{h\in M}\frac{\sigma_h(\sigma_h-1)}{2} \leqslant \frac{kT(T-1)}{2} + \frac{(|V|-k)(T-1)(T-2)}{2}.$$

We claim there is a one-to-one correspondence between the ISP on G and the CNP on $\bar{G}$. If there is an independent set of size k in G, it is clearly a solution to the CNP, as we will have k components of size T and $|V|-k$ components of size $T-1$. This would result in the required objective function value for the CNP. Conversely, if there is a partition of M satisfying the objective by deleting $|V|-k$ nodes, then we have an independent set of size k.
We give a constructive proof to show this. The first part involves showing that the objective value of the CNP on $\bar{G}$ is always better when the nodes of the original graph G are deleted from $\bar{G}$ as opposed to deleting clique nodes. For a given CNP solution, let us assume that in a clique, a non-coinciding node is deleted and the coinciding node from this clique is not deleted. If we swap these nodes in the solution, that is, if we delete the coinciding node and replace the non-coinciding node, then the objective function value either remains the same, or decreases if the number of components increases.
Now let us assume that a non-coinciding clique node and its coinciding node are deleted from $\bar{G}$. In this case, if we swap from the solution set the non-coinciding node with an undeleted coinciding node, then again the objective value will either decrease or remain unchanged. To see this, let us assume the component corresponding to the clique with both its coinciding and non-coinciding nodes deleted is of size $(T-b)$, where $b \geqslant 2$, as there may be other non-coinciding nodes deleted from this clique. Also, let the coinciding node that was not deleted be a part of some component of size $T+a$. Now the objective function before the swap will be

$$Z_1 = S + \frac{(T+a)(T+a-1)}{2} + \frac{(T-b)(T-b-1)}{2}, \tag{18}$$

where S is the contribution from other components present in the graph. After swapping these two nodes, the objective function value would be

$$Z_2 = S + \frac{(T-1)(T-2)}{2} + \frac{(a)(a-1)}{2} + \frac{(T-b)(T-b+1)}{2}. \tag{19}$$

Now, if we take the difference we see that

$$Z_1 - Z_2 = aT + b - 1 \geqslant 0, \quad \forall T \geqslant 0. \tag{20}$$

Since we are deleting only $|V|-k$ nodes, we have a partition M with

$$\sum_{h\in M}\frac{\sigma_h(\sigma_h-1)}{2} \leqslant \frac{kT(T-1)}{2} + \frac{(|V|-k)(T-1)(T-2)}{2},$$

by deleting only the nodes of the original graph. Since none of the new nodes (i.e., the nodes from the T-cliques) are deleted from $\bar{G}$, the deletion of the $|V|-k$ nodes results in exactly $|V|-k$ components of size $T-1$. This contributes exactly $((|V|-k)(T-1)(T-2))/2$ towards the objective function. The remaining kT nodes form at most k components. Hence from Lemma 2, this contributes at least $(kT(T-1))/2$ to the objective function. From Lemma 1, the kT nodes involve exactly k components of size T, representing the T-cliques of $\bar{G}$, with one node in each T-clique present in the original graph G, and none of them connected to each other. Hence deletion of $|V|-k$ nodes from $\bar{G}$ results in k independent nodes in the original graph G. This completes the proof. □

2.2. IP formulations

When studying combinatorial problems, IP models are usually quite helpful for providing some of the formal properties of the problem [11]. With this in mind we now develop a linear IP formulation for the CNP.

To begin with, define the surjection $u : V \times V \mapsto \{0,1\}$ as above. Further, we introduce a surjection $v : V \mapsto \{0,1\}$ defined by

$$v_i := \begin{cases} 1, & \text{if node } i \text{ is deleted in the optimal solution,} \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Then the CNP admits the following IP formulation

$$\text{(CNP-1)} \quad \text{Min} \sum_{i,j \in V} u_{ij} \tag{22}$$

$$\text{s.t.} \quad u_{ij} + v_i + v_j \geqslant 1, \quad \forall (i,j) \in E, \tag{23}$$

$$u_{ij} + u_{jk} - u_{ki} \leqslant 1, \quad \forall (i,j,k) \in V, \tag{24}$$

$$u_{ij} - u_{jk} + u_{ki} \leqslant 1, \quad \forall (i,j,k) \in V, \tag{25}$$

$$-u_{ij} + u_{jk} + u_{ki} \leqslant 1, \quad \forall (i,j,k) \in V, \tag{26}$$

$$\sum_{i \in V} v_i \leqslant k, \tag{27}$$

$$u_{ij} \in \{0,1\}, \quad \forall i,j \in V, \tag{28}$$

$$v_i \in \{0,1\}, \quad \forall i \in V. \tag{29}$$

Note the objective is to find the set of k nodes whose removal results in a graph which has the minimum pair-wise connectivity between the remaining nodes. This is accomplished by the objective function. The first set of constraints in (23) implies that if nodes i and j are in different components and if there is an edge between them, then one of them must be deleted. Furthermore, constraints (24)–(26) together imply that for all triplets of nodes i,j,k, that if i and j are in same component and j and k are in same component, then necessarily k and i must be in the same component. Constraint (27) ensures that the total number of deleted nodes is less than or equal to k. Finally, (28) and (29) define the proper domains for the variables used. Thus, a solution to the IP formulation CNP-1 characterizes a feasible solution to the CNP. On the other hand, it is clear that a feasible solution to the CNP will define at least one feasible solution to CNP-1. Therefore, CNP-1 is a correct formulation for the CNP.

We note here that in all likelihood, there exist alternative mathematical programming formulations for the CNP. For example, notice that the conditions which satisfy the circular constraints (24), (25), and (26) in CNP-1 can be satisfied by the single constraint $u_{ij} + u_{jk} + u_{ki} \neq 2, \forall (i,j,k) \in V$. By appropriately defining a new set of binary variables, this constraint set could be incorporated into the model. This might be useful for breaking down the symmetry of the problem if one was attempting to exploit the polyhedral structure of the model. This is beyond the scope of this paper, and is an interesting topic for future research. That said, we make no claims as to the superiority of formulation CNP-1 over others.

Notice that if the objective was a function of the number of components, then an approximation for the MAXIMUM K-CUT PROBLEM [15,16] could be employed by modifying the cost function of the Gomory-Hu tree [17]. An even simpler approach would be to identify the cut vertices in the graph, if any exist. Studies to assess the vulnerability of a network with similar objective functions are studied in [18,19]. The objective function in [19] is to maximize traffic flow while deleting a set of k edges [19] from the graph. In [18], a very similar objective to the one proposed in this paper is presented. Whereas our objective is to minimize the pair-wise connectivity between the nodes after deleting k nodes, the objective in [18] maximizes the node disconnectivity between a set of source nodes and sink nodes by deleting a set of k arcs.

Recall that $\sum_{i,j \in V} u_{ij}$ is a measure of the total pair-wise connectivity of the graph. Notice that since u_{ij} is binary and equal to 1 if and only if i and j are in the same component in the optimal solution, the objective function could be rewritten as

$$\sum_{h \in M} \frac{\sigma_h(\sigma_h - 1)}{2}, \tag{30}$$

where M is the set of all maximal connected components and σ_h is the size of the hth component, which can be easily identified by fast algorithms like breadth or depth first search algorithms in $\mathcal{O}(|E|)$ time using an adjacency list representation of the network [13,20]. We will use (30) as the objective function optimized by the heuristic in the following section. In addition to the relative ease of calculating the cardinality of the components of a graph, there is an intuitive explanation for the choice of (30) as our objective function. As we proved in Lemmas 1 and 2 above, optimizing (30) maximizes the number of connected components while simultaneously minimizing the variance in the component sizes. For example, consider an arbitrary unweighted graph with 150 nodes. According to our objective, it is more preferable to have a partition with three components each with 50 nodes, as opposed to a partition with five components with one having 146 nodes and the rest of them having a single node.

3. Heuristic for detecting critical nodes

Pseudo-code for the proposed heuristic is provided in Fig. 1. To begin with, the algorithm finds a maximal independent set (MIS). This set is initially empty, and is computed sequentially as follows. First, a single vertex is added to the set. Next by iterating through the vertices, a node that is not adjacent to the starting node is added to the MIS. Then a vertex adjacent to neither of these is added, and so on. This is continued until we can find no more vertices to include, and thus the set is maximal independent.

Let M_j be the set of all connected components in the node induced subgraph $G(MIS \cup j)$. After the initial MIS is computed, in the loop from lines 2–5, the heuristic greedily selects the node $i \in V$ not currently in the MIS which returns the minimum objective function for the graph $G(\text{MIS} \cup \{j\}), \forall j \in V$. The set MIS is augmented to include node i, and the process repeats until $|\text{MIS}| = |V| - k$. At this time, the method terminates and the set of critical nodes to be deleted is given as those nodes $j \in V$ such that $j \in V \setminus \text{MIS}$.

The intuition behind using an independent set is that the subgraph induced by this set is empty. Stated otherwise, the deletion of those nodes that are *not* in the independent set will result in an empty subgraph. Notice that this will provide the optimal solution for an instance of the CNP if $|\text{MIS}| \geqslant |V| - k$. However, if the size of

```
procedure CriticalNode$(G,k)$
1    MIS $\leftarrow$ MaximalIndepSet$(G)$
2    while $(|\text{MIS}| \neq |V| - k)$ do
3        $i \leftarrow \arg\min\{\sum_{h \in M_j} \frac{\sigma_h(\sigma_h - 1)}{2} : j \in V \setminus \text{MIS}\}$
4        MIS $\leftarrow$ MIS $\cup \{i\}$
5    end while
6    return $V \setminus$ MIS    /* set of $k$ nodes to delete */
end procedure CriticalNode
```

Fig. 1. Heuristic for detecting critical nodes.

```
procedure LocalSearch($V \setminus$ MIS)
1   $X^* \leftarrow$ MIS
2   $local\_improvement \leftarrow$ .TRUE.
3   while $local\_improvement$ do
4      $local\_improvement \leftarrow$ .FALSE.
5      for $[i,j] \in VxV$ do
6         if $i \in$ MIS and $j \notin$ MIS then
7            MIS $\leftarrow$ MIS $\setminus i$
8            MIS $\leftarrow$ MIS $\cup j$
9            if $f$(MIS) $< f(X^*)$ then
10              $X^* \leftarrow$ MIS
11           else
12              MIS $\leftarrow$ MIS $\setminus j$   /* undo swap */
13              MIS $\leftarrow$ MIS $\cup i$
14           end if
15           if $improvement\_condition\_notmet$ then
16              $local\_improvement \leftarrow$ .TRUE.
17           end if
18        end if
19     end
20  end while
21  return $(V \setminus X^*)$   /* set of $k$ nodes to delete */
end procedure LocalSearch
```

Fig. 2. Local search algorithm for critical node heuristic.

the MIS is less than $|V| - k$, we simply keep adding nodes which provide the best objective value to the set until it reaches the desired size. The heuristic is computationally efficient and the complexity is given in the following theorem.

Theorem 2. *The proposed algorithm has complexity* $\mathcal{O}(|V|^2|E|)$.

Proof. To begin with, finding the MIS using the sequential method described above requires linear time. Next, the *while* loop from lines 2–5 will iterate at most $\mathcal{O}(|V| - k)$ times. In each iteration, the number of search operations decreases from $|V| - 1$ to $|V| - (|V| - k) = k$. Note that we are performing the search of a sparse graph, which is initially empty. There will be one comparison step for every search performed in order to determine the node that provides the minimum increase in the objective function. This will in turn be dominated by the complexity of the search procedure which requires $\mathcal{O}(|E|)$ time. Hence, the total number of iterations will be

$$\mathcal{O}(|V| - 1 + |V| - 2 + \cdots + |V| - |V| + k)$$
$$= \mathcal{O}\left(\sum_{i=1}^{|V|-1} i - \sum_{i=1}^{k-1} i\right) = \mathcal{O}(|V|^2 - k^2) = \mathcal{O}(|V|^2).$$

Thus the overall complexity is $O(|V|^2|E|)$, and the proof is complete. □

The proposed algorithm finds a feasible solution to the CNP; however, the solution is not guaranteed to be globally or locally optimal. Therefore, we can enhance the heuristic with the application of a local search routine as follows. Consider the pseudo-code presented in Fig. 2. The routine receives as input the solution from the `CriticalNode` heuristic and performs a 2-exchange local search. Let $f: V \mapsto \mathbb{Z}$ be a function returning the objective function value for a given set in the sense of (30) above. That is, consider a pair of nodes i and j such that $i \in$ MIS and $j \notin$ MIS. Then for all such pairs, we set $j \in$ MIS and $i \notin$ MIS and examine the change in the objective function. If it improves, then the swap is kept; otherwise, we undo the swap and continue to the next node pair. Notice that the loop from lines 3–20 repeats while the local improvement condition is not met. This general statement can lead to implementation problems and it is a common practice to limit the number of local search iterations by some user defined value. The intuition is that as this value grows larger, the solution gets closer to optimality with respect to its local neighborhood.

```
procedure CriticalNodeLS($G, k$)
1   $X^* \leftarrow \emptyset$
2   $f(X^*) \leftarrow \infty$
3   for $j = 1$ to MaxIter do
4      $X \leftarrow$ CriticalNode($G, k$)
5      $X \leftarrow$ LocalSearch(X)
6      if $f(X) < f(X^*)$ then
7         $X^* \leftarrow X$
8      end if
9   end
10  return $(V \setminus X^*)$   /* set of $k$ nodes to delete */
end procedure CriticalNodeLS
```

Fig. 3. Heuristic with local search for detecting critical nodes.

Finally, we can combine the construction and local improvement algorithms into one multi-start heuristic `CriticalNodeLS` as shown in Fig. 3. This procedure produces `MaxIter` local optima and the overall best solution from all iterations is returned. In order to implement the multi-start framework, the starting node for each MIS is randomly chosen. Since the initial MIS is created deterministically, this node is only accepted as a starting node if it has not been previously selected. Therefore, we see that `MaxIter` will be bounded above by $|V|$. This simple randomization scheme ensures that different areas of the solution space are explored in each iteration.

4. Computational results

The proposed heuristic was implemented in the C++ programming language and compiled using GNU g++ version 3.4.4, using optimization flags -O2. It was tested on a PC equipped with a 1700 MHz Intel® Pentium® M processor and 1.0 gigabytes of RAM operating under the Microsoft® Windows® XP Professional environment. The parameter `MaxIter` was set equal to $|V|$, and the number of iterations of the local search was 2. It is reasonable to forego the implementation of the local search procedure using this approach, and simply allow the routine to examine swaps until a local improvement condition is met. This local improvement condition could be a maximum computation time, a maximum number of iterations in which there is no improvement in the objective value, or a fixed number of iterations. In our experiments, we obtained good results in reasonable computing times by fixing the number of local search iterations to 2.

As a basis of comparison, we have implemented the IP model for the CNP using the CPLEX™ version 9 optimization suite from ILOG [21]. CPLEX contains an implementation of the simplex method [22], and uses a branch-and-bound algorithm [23] together with advanced cutting-plane techniques [24,25].

We tested the IP model and heuristic on a set of randomly generated graphs ranging in size from 75 to 150 nodes with varying densities. The graphs were generated with version 1.4 of the publicly available Barabási graph generator by Dreier [26]. For each random instance, we report solutions for 3 values of k, the number of nodes to be deleted. In addition, we have tested the algorithms on the terrorist network compiled by Krebs [7] shown in Fig. 4. This network depicts the relationships between the terrorists involved in the horrific attacks of September 11, 2001. The graph was constructed after the attacks with data which were publicly available before 9/11.

We begin by providing the results from the terrorist network [7] shown in Fig. 4. This graph has 62 nodes and 153 edges. Notice that

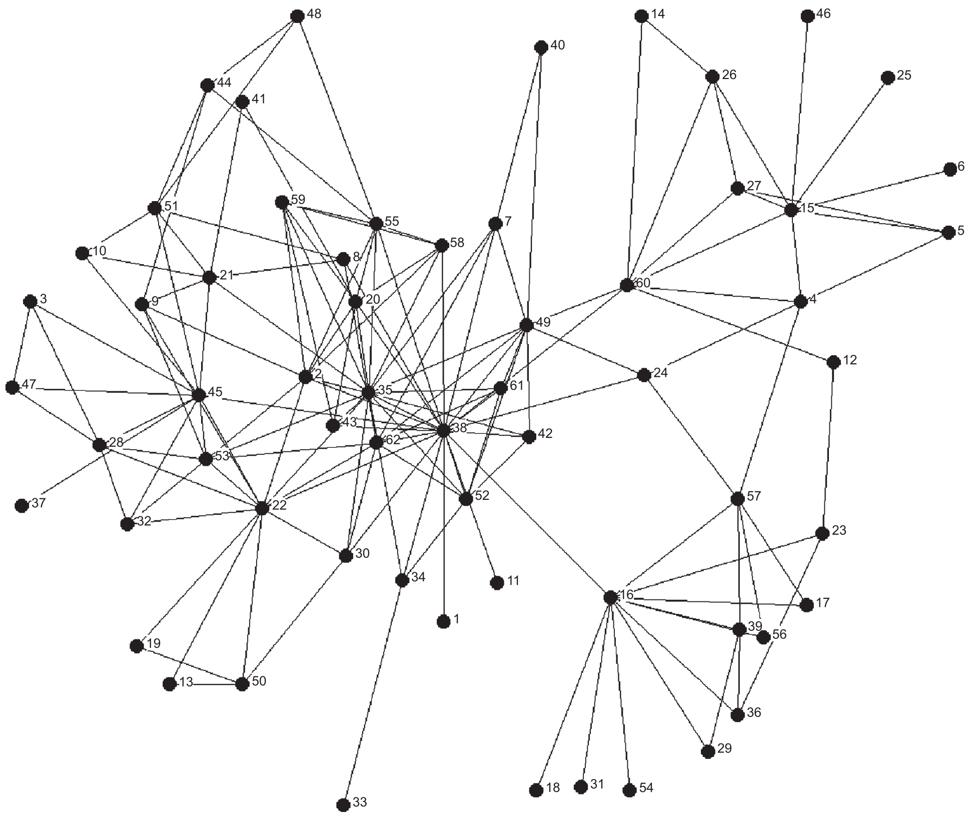

Fig. 4. Terrorist network compiled by Krebs.

Table 1
Results of IP model and heuristic on terrorist network data from Krebs.

Instance	IP model		Heuristic	
Nodes deleted (k)	Objective value	Execution time (s)	Objective value	Execution time (s)
20	20	12.69	20	0.01
15	61	277.77	61	0.01
10	169	3337.06	169	0.02
9	214	2792.33	214	0.02
8	282	15 111.94	282	0.01
7	327	10 792.08	327	0.01

node 38 is the central node with degree 22. We applied the IP formulation and the heuristic to this network with 6 values of k. The results are provided in Table 1. Notice that for all values of k, the heuristic computed the optimal solution requiring on average 0.013 s of computation time. The average time to compute the optimal solution using CPLEX was 5387.31 s. Clearly even for this relatively small network, the heuristic is the method of choice. Fig. 5 shows the resulting graph of the terrorist network according to the optimal solution to the CNP for the instance with $k = 20$.

In order to provide evidences of its scalability and robustness, the proposed heuristic was tested on a set of randomly generated scale-free graphs. Table 2 presents the results of the heuristic and the optimal solver when applied to the random instances. For each instance, we report the number of nodes and arcs, the value of k being considered, the optimal solution and computation time, and finally the heuristic solution and the corresponding computation time. For each graph, we report solutions for three different values of k.

Notice that for all the instances tested, our method was able to compute the optimal solution. Furthermore, the required time to compute the optimal solution was less than one second for all but one instance, averaging only 0.33 s for all 27 instances. On the other hand, CPLEX required 289.44 s on average to compute the optimal solution, requiring over 5000 s in the worst case. Our computational experiments indicate that the proposed heuristic is able to efficiently provide excellent solutions for large-scale instances of the CNP.[1]

5. Conclusion

In this paper, we propose a novel approach for identifying the critical nodes of a sparse graph, whose deletion results in maximum network disconnectivity. This problem has a wide variety of applications from epidemiology to anti-terrorism protection. The proposed method is based on integer linear programming. In addition, we prove that the corresponding decision problem is $\mathcal{NP}$-complete. Furthermore, we describe the implementation of a heuristic for efficiently computing solutions to large-scale instances. The heuristic

[1] The experimental data including the instances tested are available at the following URL: http://plaza.ufl.edu/clayton8/cnp/.

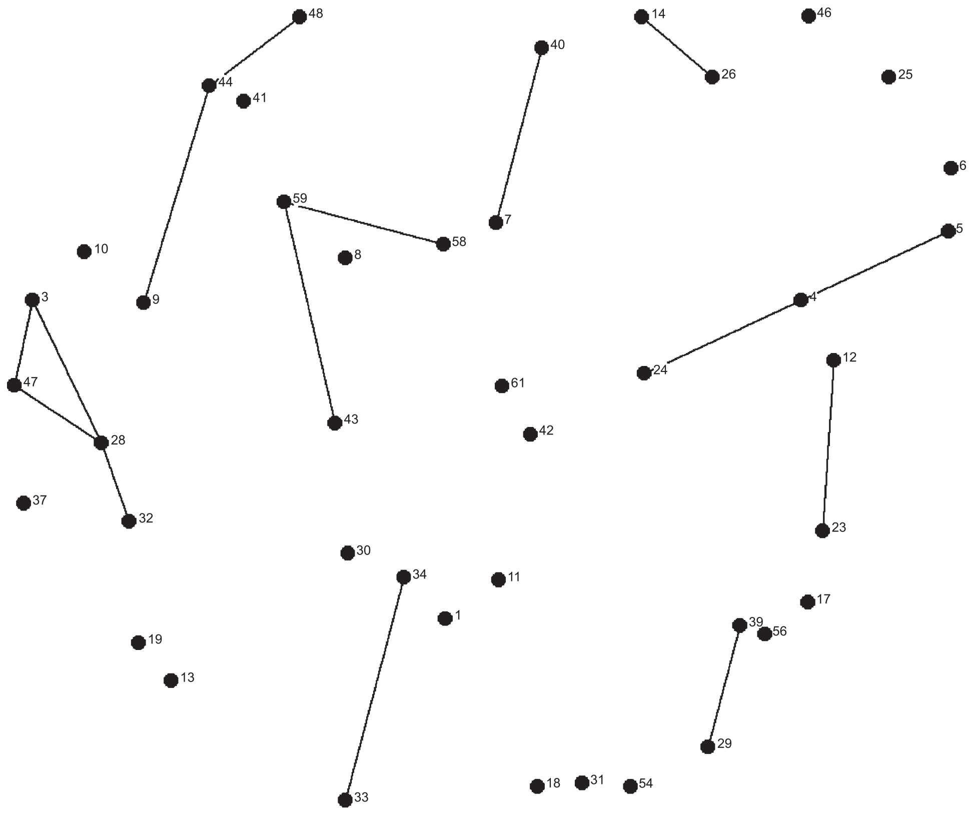

Fig. 5. Optimal solution when $k=20$.

Table 2
Results of IP model and heuristic on randomly generated scale free graphs.

Instance			IP model		Heuristic	
Nodes	Arcs	Deleted nodes (k)	Objective value	Execution time (s)	Objective value	Execution time (s)
75	140	20	36	66.7	36	0.03
75	140	25	18	33.28	18	0.03
75	140	30	7	4.23	7	0.04
75	210	25	26	93.71	26	0.04
75	210	30	8	3.57	8	0.05
75	210	35	2	4.36	2	0.04
75	280	33	26	749.19	26	0.04
75	280	35	20	164.34	20	0.06
75	280	37	13	83.98	13	0.11
100	194	25	44	151.14	44	0.09
100	194	30	20	59.66	20	0.11
100	194	35	10	8.51	10	0.12
100	285	40	23	136.47	23	0.11
100	285	42	17	263.82	17	0.17
100	285	45	11	16.78	11	0.23
100	380	45	22	128.13	22	0.15
100	380	47	16	243.07	16	0.16
100	380	50	10	228.72	10	0.11
125	240	33	62	5047.51	62	0.30
125	240	40	29	118.92	29	0.24
125	240	45	16	17.09	16	0.39
150	290	40	40	41.6	40	0.47
150	290	50	12	26.29	12	0.831
150	290	60	1	24.92	1	0.851
150	435	61	19	29.55	19	0.741
150	435	65	13	31.45	13	1.952
150	435	67	11	37.91	11	0.801

is further intensified by the application of a local search mechanism. Computational results indicate that the heuristic produces high quality solutions in a fraction of the time required by the commercial integer programming solver CPLEX.

Acknowledgments

The authors sincerely thank the anonymous reviewer whose insightful comments and suggestions greatly improved the quality of this paper. In addition, the authors gratefully acknowledge the Air Force Office of Scientific Research and the National Science Foundation for providing financial support for this research.

References

[1] Bavelas A. A mathematical model for group structure. Human Organizations 1948;7:16–30.
[2] Freeman LC. Centrality in social networks I: conceptual clarification. Social Networks 1979;1:215–39.
[3] Albert R, Barabási A-L. Emergence of scaling in random networks. Science 1999;286:509–12.
[4] Resende MGC, Pardalos PM. Handbook of optimization in telecommunications. Berlin: Springer; 2006.
[5] Cohen R, Ben Avraham D, Havlin S. Efficient immunization strategies for computer networks and populations. Physical Review Letters 2003;91: 247901–5.
[6] Zhou T, Fu Z-Q, Wang B-H. Epidemic dynamics on complex networks. Progress in Natural Science 2006;16:452–7.
[7] Krebs V. Uncloaking terrorist networks, First Monday, 7, 2002.
[8] Commander CW, Pardalos PM, Ryabchenko V, Uryasev S, Zrazhevsky G. The wireless network jamming problem. Journal of Combinatorial Optimization 2007;14:481–98.
[9] Elefteriadou L. Highway capacity. In: Kutz M, editor. Handbook of transportation engineering. New York: McGraw-Hill; 2004. p. 8-1–8-17 (chapter 8).
[10] Borgatti SP. Identifying sets of key players in a network. Computational and Mathematical Organization Theory 2006;12:21–34.
[11] Oliveira CAS, Pardalos PM, Querido TM. Integer formulations for the message scheduling problem on controller area networks. In: Grundel D, Murphey R, Pardalos P, editors. Theory and algorithms for cooperative systems. Singapore: World Scientific; 2004. p. 353–65.
[12] Narayanan H, Roy S, Patkar S. Approximation algorithms for min-k-overlap problems using the principal lattice of partitions approach. Journal of Algorithms 1996;21:306–30.
[13] Ahuja RK, Magnanti TL, Orlin JB. Network flows: theory, algorithms, and applications. Englewood Cliffs, NJ: Prentice-Hall; 1993.
[14] Butenko SI. Maximum independent set and related problems with applications, PhD thesis, University of Florida, 2003.
[15] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. New York: W.H. Freeman and Company; 1979.
[16] Karp R. Reducibility among combinatorial problems. In: Proceedings of a symposium on the complexity of computer computations. 1972.
[17] Gomory RE, Hu TC. Multi-terminal network flows. Journal of SIAM 1961;9: 551–70.
[18] Matisziw TC, Murray AT. Modeling s–t path availability to support disaster vulnerability assessment of network infrastructure. Computers and Operations Research 2009;36(1):16–26.
[19] Myung YS, Kim HJ. A cutting plane algorithm for computing k-edge survivability of a network. European Journal of Operational Research 2004;156:579–89.
[20] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. Cambridge MA: MIT Press; 2001.
[21] ILOG CPLEX ⟨http://www.ilog.com/products/cplex⟩, Accessed October 2006.
[22] Hillier FS, Lieberman GJ. Introduction to operations research. New York: McGraw-Hill; 2001.
[23] Wolsey L. Integer programming. New York: Wiley; 1998.
[24] Horst R, Pardalos PM, Thoai NV. Introduction to global optimization nonconvex optimization and its applications, vol. 3. Dordrecht: Kluwer Academic Publishers; 1995.
[25] Oliveira CAS, Pardalos PM, Querido TM. A combinatorial algorithm for message scheduling on controller area networks. International Journal of Operations Research 2005;1:160–71.
[26] Dreier D. Barabasi graph generator v1.4 ⟨http://www.cs.ucr.edu/ddreier⟩, Accessed November 2006.