MANAGING NETWORK RISK VIA CRITICAL NODE IDENTIFICATION

ASHWIN ARULSELVAN, CLAYTON W. COMMANDER, PANOS M. PARDALOS, AND OLEG SHYLO

ABSTRACT. We consider methodologies for managing risk in a telecommunication network based on identification of the critical nodes. The objective is to identify a set of vertices with a specified cardinality whose deletion results is maximum number of disconnected components. This is referred to as the CRITICAL NODE PROBLEM, and finds application in epidemic control, telecommunications, and military tactical planning, among others. From a telecommunication perspective, the set of critical nodes helps determine which players should be removed from the network in the event of a virus outbreak. Conversely, in order to maintain maximum global connectivity, it should be ensured that the critical nodes remain intact. In this chapter, we review the recent work in this area and examine formulations based on integer linear programming.

1. INTRODUCTION

In this chapter, we study two variants of the CRITICAL NODE PROBLEM. In general, the objective of the CRITICAL NODE PROBLEM (CNP) is to find a set of k nodes in a graph whose deletion results in the maximum network fragmentation. By this we mean, maximize the number of components in the k-vertex deleted subgraph. Studies carried out in this line include those by Bavelas [6] and Freeman [15] which emphasize node centrality and prestige, both of which are usually functions of a nodes degree. However, they lacked applications to problems which emphasized network fragmentation and connectivity.

We can apply the CNP to the problem of jamming wired telecommunication networks by identifying the critical nodes and suppressing the communication on these nodes. This will result in the maximum number of disconnected components which are unable to communicate with each other. The CNP can also be applied to the study of covert terrorist networks, where a certain number of individuals have to be identified whose deletion would result in the maximum breakdown of communication between individuals in the network [24]. 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 offline.

The CNP also finds applications in network immunization [9, 34] 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. [9, 34]) offering control of epidemic outbreaks; however, none of the proposed are optimal strategies. The vaccination strategies suggested emphasize the *centrality* of nodes as a major factor rather than *critical* nodes whose deletion will maximize 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. Of course, owing to its dynamic stature, the relationships between people, represented by edges in the social network are transient and there is a constant rewiring between nodes, and alternate relationships could

Date: April 2007.

Air Force Research Laboratory Technical Report #:

Submitted to *Risk Management in Telecommunication Networks*, B. Rustem and N. Gulpinar (editors), Springer, 2007.

be established in the future. The proposed critical node technique helps in a maximum prevention of disease transmission over an instance of the dynamic network.

Before proceeding, we mention one final area in which the CRITICAL NODE PROBLEM finds several applications, and that is in the field of transportation engineering [14]. 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 [7] 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 [6, 15] 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 because 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 [27]. In the next section, we present a mathematical model based on integer linear programming which provides optimal solutions for the CRITICAL NODE PROBLEM.

We organize this chapter 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 to large-scale 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 determined using a commercial software package. Some concluding remarks are given in Section 5.

2. PROBLEM FORMULATIONS

Denote a graph $G = (V, E)$ as a pair consisting of a set of vertices V, and a set of edges E. All graphs in this chapter are assumed to be undirected and unweighted. For a subset $W \subseteq V$, let $G(W)$ denote the subgraph induced by W on G. A set of vertices $I \subseteq V$ is called an *independent* or *stable set* if for every $i, j \in I,\ (i, j) \notin E$. That is, the graph $G(I)$ induced by I is edgeless. An independent set is *maximal* if it is not a subset of any larger independent set (*i.e.*, it is maximal by inclusion), and *maximum* if there are no larger independent sets in the graph.

2.1. Critical Node Problem.

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: $A = \arg\min \sum_{i,j \in (V \setminus A)} u_{ij}\big(G(V \setminus A)\big) : |A| \leq k$, where

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

The objective is to find a subset $A \subseteq V$ of nodes such that $|A| \leq k$, whose deletion results in the minimum value of $\sum u_{ij}$ in the edge induced subgraph $G(V \setminus A)$. This objective function results in a minimum cohesion in the network, while also ensuring a minimum difference in the sizes of the components. An illustration is best suited to explain the choice of objective function. Consider an arbitrary unweighted graph with 150 nodes. According to our objective, it is more preferable to have a partition with 3 components with

each 50 nodes as opposed to a partition with 5 components with one having 146 nodes and the rest of them having a single node.

This problem is similar to MINIMUM k-VERTEX SHARING [26], 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 [7] has given a comprehensive illustration to facilitate the understanding of the objective function and its non-triviality.

We now prove that the recognition version of the CNP is $\mathcal{NP}$-complete. Consider the following decision problem for the CNP:

K-CRITICAL NODE PROBLEM (K-CNP)
INPUT: An undirected graph $G = (V, E)$ and an integer k.
QUESTION: Does there exist a zero cost K-way partition of G by deleting k nodes or less?

Theorem 1. *The* K-CRITICAL NODE PROBLEM *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 $G = (V, E)$, we can verify the validity of G in polynomial time. More specifically, by deleting any set of at most k nodes, we determine if the there is a zero-cost K-way partition of G in $\mathcal{O}(|E| + |V|)$ time using a depth-first search [1].

(2) To complete the proof, we show a reduction from the K-INDEPENDENT SET PROBLEM (K-ISP) [8], which is well-known to be $\mathcal{NP}$-complete [16]. Recall that the objective of the K-ISP is to determine if G contains an independent set containing at least K nodes. Let $G = (V, E)$ be a graph in which we seek an independent set. There are no necessary transformations required for the graph in which we are solving the corresponding K-CNP. We will show that a 'yes' instance of the K-ISP corresponds to a 'yes' instance of the K-CNP on G. In particular, G has an independent set of size K if and only if the K-CNP has a zero cost solution where $k \leq |V| - K$. Suppose G contains an independent set I where $|I| = K$. Notice that the objective of the K-CNP will be 0 as the subgraph induced by deleting the nodes in $V \setminus I$ is edgeless. Therefore, a 'yes' instance of the K-ISP implies a 'yes' instance for the K-CNP with $k = |V| - K$.

To prove the converse, observe that the cost of any K-CNP is at least 0. Thus, a 'yes' instance of the K-CNP would imply that once the k critical nodes are removed, the resulting subgraph consists of K components whose objective function is 0. This implies that the induced subgraph is edgeless, i.e. each of the K components consists of a single node. Hence, the K remaining nodes form an independent set of G, resulting in a 'yes' instance for the K-INDEPENDENT SET PROBLEM. Thus the proof is complete.

□

When studying combinatorial problems, integer programming models are usually quite helpful for providing some of the formal properties of the problem [27]. With this in mind we now develop a linear integer programming 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

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

Then the CRITICAL NODE PROBLEM admits the following integer programming formulation

$$\textbf{(CNP-1)} \quad \text{Minimize} \quad \sum_{i,j \in V} u_{ij} \tag{2}$$

s.t.

$$u_{ij} + v_i + v_j \geq 1, \ \forall \ (i,j) \in E, \tag{3}$$
$$u_{ij} + u_{jk} - u_{ki} \leq 1, \ \forall \ (i,j,k) \in V, \tag{4}$$
$$u_{ij} - u_{jk} + u_{ki} \leq 1, \ \forall \ (i,j,k) \in V, \tag{5}$$
$$-u_{ij} + u_{jk} + u_{ki} \leq 1, \ \forall \ (i,j,k) \in V, \tag{6}$$
$$\sum_{i \in V} v_i \leq k, \tag{7}$$
$$u_{ij} \in \{0,1\}, \ \forall \ i,j \in V, \tag{8}$$
$$v_i \in \{0,1\}, \ \forall \ i \in V. \tag{9}$$

Theorem 2. ***CNP-1*** *is a correct formulation for the* CRITICAL NODE PROBLEM.

Proof. First, we note that the objective is to find the set of k nodes whose removal results in a graph which has the maximum number of disconnected components. This is accomplished by the objective function. Notice that the first set of constraints in (3) 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 (4)-(6) 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 k and i must be in the same component. Constraint (7) ensures that the total number of deleted nodes is less than or equal to k. Finally, (8) and (9) define the proper domains for the variables used. Thus, a solution to the integer programming 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. □

Notice that the conditions which satisfy the circular constraints (4), (5), and (6) in **CNP-1** can be satisfied by the single constraint $u_{ij} + u_{jk} + u_{ki} \neq 2, \forall \ (i,j,k) \in V$. Thus we have an equivalent, more compact integer program given as

$$\textbf{(CNP-2)} \quad \text{Minimize} \quad \sum_{i,j \in V} u_{ij} \tag{10}$$

s.t.

$$u_{ij} + v_i + v_j \geq 1, \ \forall \ (i,j) \in E, \tag{11}$$
$$u_{ij} + u_{jk} + u_{ki} \neq 2, \ \forall \ (i,j,k) \in V, \tag{12}$$
$$\sum_{i \in V} v_i \leq k, \tag{13}$$
$$u_{ij} \in \{0,1\}, \ \forall \ i,j \in V, \tag{14}$$
$$v_i \in \{0,1\}, \ \forall \ i \in V, \tag{15}$$

where $u_{i,j}$ and v_i are as defined above.

Notice that if the objective function had only the number of components, then an approximation for the MAXIMUM K-CUT PROBLEM [16, 23] could be employed by modifying the cost function of the Gomory-Hu tree [19]. An even simpler approach would be to identify the cut vertices in the graph, if any exist. However, the objective function also involves the sizes of the components formed, which makes the problem harder and subsequently implies that the methods suggested above are not suitable for our problem.

Recall that $\sum_{i,j \in V} u_{ij}$ is a measure of the total disconnectivity of the graph. If we observe carefully, the objective function could be rewritten as

$$\sum_{i \in S} \frac{s_i(s_i - 1)}{2}, \tag{16}$$

where S is set of all components and s_i is the size of the ith component, which can be easily identified by fast algorithms like breadth or depth first search algorithms in $\mathcal{O}(|V| + |E|)$ time [11]. We now provide an intuitive explanation for the choice of our objective function. For a fixed number of components the variance in the sizes of the components will be the sum of the squares of deviation of sizes of the components from the mean size of a component. However notice that the mean size of any component is constant because the sum of the sizes of the components is the constant, $|V| - k$. Thus minimizing the variance of the size of the components reduces to minimizing the sum of squares of the sizes of the components, which is our objective function. Also, when the sizes of the components are equal the objective function is the minimum when the number of components is the maximum. We will use this objective function in the following section to implement a heuristic for identifying critical nodes.

2.2. **Cardinality Constrained Problem.** We now provide the formulation for a slightly modified version of the CNP based on constraining the connectivity index of the nodes in the graph. Given a graph $G = (V, E)$, the *connectivity index* of a node is defined as the

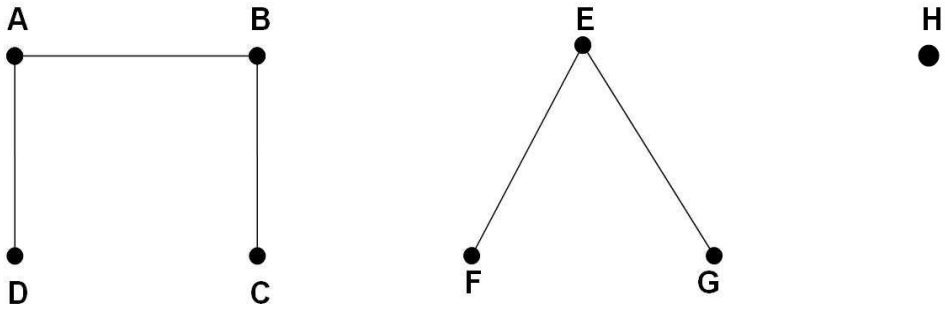

FIGURE 1. Connectivity Index of nodes A,B,C,D is 3. Connectivity Index of E,F,G is 2. Connectivity Index of H is 0.

number of nodes reachable from that vertex (see Figure 1 for examples). To constrain the network connectivity in optimization models, we can impose constraints on the connectivity indices.

This leads to a cardinality constrained version of the CNP which we aptly refer to as the CARDINALITY CONSTRAINED CRITICAL NODE DETECTION PROBLEM (CC-CNP). The objective is to detect a set of nodes $A \subseteq V$ such that the connectivity indices of the nodes in the vertex deleted subgraph $G(V \setminus A)$ is less than some threshold value, say L. Using the same definition of the variables as in the previous subsection, we can formulate the

CC-CNP as the following integer linear programming problem.

$$\textbf{(CC-CNP-1)} \quad \text{Minimize} \quad \sum_{i \in V} v_i \tag{17}$$

s.t.

$$u_{ij} + v_i + v_j \geq 1, \ \forall \ (i,j) \in E, \tag{18}$$

$$u_{ij} + u_{jk} + u_{ki} \neq 2, \ \forall \ (i,j,k) \in V, \tag{19}$$

$$\sum_{i,j \in V} u_{ij} \leq L, \tag{20}$$

$$u_{ij} \in \{0,1\}, \ \forall \ i,j \in V, \tag{21}$$

$$v_i \in \{0,1\}, \ \forall \ i \in V, \tag{22}$$

where L is the maximum allowable connectivity index for any node in V.

Theorem 3. ***CC-CNP1*** *is a correct formulation for the* CARDINALITY CONSTRAINED CRITICAL NODE DETECTION PROBLEM.

Proof. This proof follows in much the same way as Theorem 2. First, we see that the objective function given clearly minimizes the number of nodes deleted. Constraints (18) and (19) follow exactly as in the CNP formulation. The only difference is now we must constrain the connectivity index of each node. This is accomplished by constraint (20). Finally constraints (21) and (22) define the domains of the decision variables, and we have the proof. □

3. HEURISTICS FOR CRITICAL NODE PROBLEMS

3.1. **CNP Heuristic.** Pseudo-code for the proposed heuristic is provided in Figure 2. To begin with, the algorithm finds a maximal independent set (MIS). Then in the loop from lines 2-5, the heuristic greedily selects the node $i \in V$ not currently in MIS which returns the minimum objective function for the graph $G(\text{MIS} \cup \{i\})$. The set MIS is augmented to include node i, and the process repeats until $|\text{MIS}| = |V| - k$. The method terminates and the set of critical nodes to be deleted is given as those nodes $j \in V$ such that $j \in V \setminus \text{MIS}$.

```
procedure CriticalNode(G, k)
1    MIS ← MaximalIndepSet(G)
2    while (|MIS| ≠ |V| − k) do
3       i ← arg min{ Σ_{i∈S} s_i(s_i−1)/2 : S ∈ G(MIS ∪ {i}), i ∈ V \ MIS }
4       MIS ← MIS ∪ {i}
5    end while
6    return V \ MIS    /* set of k nodes to delete */
end procedure CriticalNode
```

FIGURE 2. Heuristic for detecting critical nodes.

The intuition behind using an independent set is that the subgraph induced by this set is empty. Stated otherwise, the deletion of nodes that are *not* in the independent set from the graph will result in an empty subgraph. Notice that this will provide the optimal solution for an instance of the CNP if $|\text{MIS}| \geq |V| - k$. However, if the size of 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. In the following lemma, we establish a relationship between the CNP and the MAXIMUM INDEPENDENT SET problem, which also provides a bound on the optimal solution for an instance of the CNP.

Lemma 1. *Given a graph $G = (V, E)$, the cardinality of the maximum independent set of G, denoted $\alpha(G)$ provides an upper bound on the number of components produced in the optimal solution of the corresponding* CRITICAL NODE PROBLEM *for any value of $k \in \mathbb{Z}$.*

Proof. Obviously, removing the critical nodes determined by the optimal solution for any instance of the CNP results in a set of disconnected components of G. One node from each of these components forms an independent set. Hence $\alpha(G)$ should be at least as large as the number of components formed in the optimal solution to the CNP. Furthermore, the components formed in the subgraph induced by the maximum independent set are of size one, and hence result in the optimal solution for the CNP instance if $\alpha(G) \geq |V| - k$, i.e. if the deletion of some k nodes results in an empty graph. Thus, we have the lemma. □

We note that this bound is not particularly useful in practice since the MAXIMUM INDEPENDENT SET problem is $\mathcal{NP}$-hard in general [8, 16]. However, a maximal independent set can be computed in polynomial time. This motivates our decision to use *maximal* instead of *maximum* independent sets in the heuristic. Subsequently the heuristic is computationally efficient, with the complexity given in the following theorem.

Theorem 4. *The proposed algorithm has complexity $\mathcal{O}(k^2 + |V|k))$.*

Proof. To begin with, 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. Hence the total complexity will be

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

Thus the proof is complete. □

The proposed algorithm finds a feasible solution to the CRITICAL NODE PROBLEM; however, the solution is not guaranteed to be globally or locally optimal. Therefore, we can enhance the heuristic with the application of local search routine as follows. Consider the pseudo-code presented in Figure 3. 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 (16) above. That is, consider a pair of nodes i and j such that $i \in \text{MIS}$ and $j \notin \text{MIS}$. Then for all such pairs, we set $j \in \text{MIS}$ and $i \notin \text{MIS}$ 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 3-13 repeats while the solution is not locally optimal. 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, say U. The intuition is that the as $U \to \infty$, the solution becomes optimal with respect to its local neighborhood.

Theorem 5. *If the number of iterations of the local search is bounded by a constant $U \in \mathbb{R}$ as described above, then the complexity of the procedure is $\mathcal{O}(|V|^2 U)$.*

Proof. The is clear as the while loop from lines 3-13 will iterate U times. Since each iteration requires an examination of $|V|^2$ components, we have the proof. □

Finally, we can combine the construction and local improvement algorithms into one multi-start heuristic `CriticalNodeLS` as shown in Figure 4. This procedure produces `MaxIter` local optima and the overall best solution from all iterations is returned.

Theorem 6. *The* `CriticalNodeLS` *heuristic has overall complexity of $\mathcal{O}(|V|^2 UT(k^2 + |V|k))$, where $T =$* `MaxIter`*, and U is the iteration limit on the local search.*

```
procedure LocalSearch(V \ MIS)
1    X* ← MIS
2    local_improvement ← .TRUE.
3    while local_improvement do
4       local_improvement ← .FALSE.
5       if i ∈ MIS and j ∉ MIS then
6          MIS ← MIS \ i
7          MIS ← MIS ∪ j
8          if f(MIS) < f(X*) then
9             X* ← MIS
10            local_improvement ← .TRUE.
11         end if
12      end if
13   end while
14   return (V \ X*)    /* set of k nodes to delete */
end procedure LocalSearch
```

FIGURE 3. Local search algorithm for critical node heuristic.

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

FIGURE 4. Heuristic with local search for detecting critical nodes.

Proof. This result follows directly from Theorem 4 and Theorem 5 above. □

3.2. **CC-CNP Heuristic.** With a subtle modification to the heuristic described above for the CNP, we can create an effective heuristic for the CC-CNP. To do this, notice that now we are only concerned with the connectivity indices of the nodes. Stated differently, we are only concerned with the sizes of the components in the vertex deleted subgraph. Unlike before, there is no limit on the number of critical nodes we choose, so long as the connectivity constraints are satisfied.

Pseudo-code for the proposed algorithm is provided in Figure 5. The heuristic starts off the same as before by identifying a maximal independent set (MIS). Then, the boolean variable OPT is set to FALSE. Finally in line 3, a variable NoAdd is initialized to 0. This variable determines when to exit the main loop from lines 4-16. After this loop is entered, the procedure iterates through the vertices and determines which can be added back to the graph while still maintaining feasibility. If vertex i can be added, MIS is augmented to include i in step 7, otherwise NoAdd is incremented. If NoAdd is ever equal to $|V|-|\text{MIS}|$,

```
procedure ConstrainedCriticalNode(G, L)
1    MIS ← MaximalIndepSet(G)
2    OPT ← FALSE
3    NoAdd ← 0
4    while (OPT .NOT.TRUE) do
5      for (i = 1 to |V|) do
6        if ( |s|(|s|−1)/2 ≤ L ∀ s ∈ S ⊆ G(MIS ∪ {i}) : i ∈ V \ MIS) then
7          MIS ← MIS ∪ {i}
8        else
9          NoAdd ← NoAdd +1
10       end if
11       if (NoAdd = |V| − |MIS|) then
12         OPT ← TRUE
13         BREAK
14       end if
15     end for
16   end while
17   return V \ MIS    /* set of nodes to delete */
end procedure ConstrainedCriticalNode
```

FIGURE 5. Heuristic for the CARDINALITY CONSTRAINED CRITICAL NODE PROBLEM.

then no nodes can be returned to the graph and OPT is set to TRUE. Then loop is then exited and the algorithm returns the set of nodes to be deleted, i.e. $V \setminus$MIS.

Theorem 7. *The worst-case complexity of the* ConstrainedCriticalNode *heuristic is* $\mathcal{O}(|V|^2 + |V||E|)$.

Proof. This proof is similar to the proof of Theorem 4 above. The loop from lines 4-16 will iterate at most $\mathcal{O}(|V|)$ times. Each loop requires at most $\mathcal{O}(|V| + |E|)$ time to verify the if a solution will remain feasible after a node is re-included in the graph. Thus we have the result. □

3.3. **Genetic Algorithm for the CC-CNP.**

```
procedure GeneticAlgorithm
1    Generate population P_k
2    Evaluate population P_k
3    while terminating condition not met do
4      Select individuals from P_k and copy to P_{k+1}
5      Crossover individuals from P_k and put in P_{k+1}
6      Mutate individuals from P_k and put in P_{k+1}
7      Evaluate population P_{k+1}
8      P_k ← P_{k+1}
9      P_{k+1} ← ∅
10   end while
11   return best individual in P_k
end procedure GeneticAlgorithm
```

FIGURE 6. Pseudo-code for a generic genetic algorithm.

As mentioned in Subsection **??**, genetic algorithms (GAs) mimic the biological process of evolution. In this subsection, we describe the implementation of a GA for the CC-CNP. Recall the general structure of a GA as outlined in Figure 6. When designing a genetic algorithm for an optimization problem, one must provide a means to encode the population, define the crossover operator, and define the mutation operator which allows for random changes in offspring to help prevent the algorithm from converging prematurely [3].

For our implementation, we use binary vectors as an encoding scheme for individuals within the population of solutions. When the population is generated, (Figure 6, line 1), a random deviate from a distribution which is uniform onto $(0, 1) \in \mathbb{R}$ is generated for each node. If the deviate exceeds some specified value, the corresponding allele is assigned value 1, indicating this node should be deleted. Otherwise, the allele is given a 0, implying it is not deleted. In order to evaluate the fitness of the population, per line 2, we must determine whether each individual solution is feasible or not. Determining feasibility is a relatively straightforward task and can accomplished in $\mathcal{O}(|V| + |E|)$ using a depth-first search [1].

In order to evolve the population over successive generations, we use a reproduction scheme in which the parents chosen to produce the offspring are selected using the binary tournament method [25, 32]. Using this method, two chromosomes are chosen at random from the population and the one having the best fitness, i.e. the lowest objective function value, is kept as a parent. The process is then repeated to select the second parent. The two parents are then combined using a crossover operator to produce an offspring [20].

To breed new solutions, we implement a strategy known as *parameterized uniform crossover* [31]. This method works as follows. After the selection of the parents, refer to the parent having the best fitness as `MOM`. For each of the nodes (alleles), a biased coin is tossed. If the result is heads, then the allele from the `MOM` chromosome is chosen. Otherwise, the allele from the least fit parent, call it `DAD`, is selected. The probability that the coin lands on heads is known as `CrossProb`, and is determined empirically. Figure 7 provides an example of a potential crossover when the number of nodes is 5 and $\texttt{CrossProb} = 0.65$ [3].

Coin Toss	T	H	H	T	H
`MOM`	0.56	0.81	0.22	0.7	0.86
`DAD`	0.29	0.49	0.98	0.12	0.32
Offspring	0.29	0.81	0.22	0.12	0.86

FIGURE 7. An example of the crossover operation. In this case, $\texttt{CrossProb} = 0.65$.

After the child is produced, the mutation operator is applied. Mutation is a randomizing agent which helps prevent the GA from converging prematurely and escape to local optima. This process works by flipping a biased coin for each allele of the chromosome. The probability of the coin landing heads, known as the mutation rate (`MutRate`) is typically a very small user defined value. If the result is heads, then the value of the corresponding allele is reversed. For our implementation, $\texttt{MutRate} = 0.03$.

After the crossover and mutation operators create the new offspring, it replaces a current member of the population using the so-called *steady-state* model [10, 20, 25]. Using this methodology, the child replaces the least fit member of the population, provided that a clone of the child is not an existing member in the population. This method ensures that the worst element of the population is monotonically improving in every generation. In the subsequent iteration, the child becomes eligible to be a parent and the process repeats. Though the GA does converge in probability to the optimal solution, it is common to stop the procedure after some "terminating condition" (see Figure 6, line 3) is satisfied. This condition could be one of several things including, a maximum running time, a target

objective value, or a limit on the number of generations. For our implementation, we use the latter option and the best solution after `MaxGen` generations is returned.

4. COMPUTATIONAL RESULTS

All of the proposed heuristics were implemented in the `C++` programming language and complied using GNU `g++` version $3.4.4$, using optimization flags `-O2`. It was tested on a `PC` equipped with a 1700MHz Intel® Pentium® M processor and 1.0 gigabytes of RAM operating under the Microsoft® Windows® XP Professional environment.

4.1. **CNP Results.**

We begin with the numerical results of the combinatorial algorithm for the CRITICAL NODE PROBLEM. We tested the IP model and the aforementioned heuristic on the terrorist network from Krebs [24] as well as on a set of randomly generated scale-free [5] graphs ranging in size from 75 to 150 nodes with various densities. The graphs were generated with version 1.4 of the publicly available Barabási graph generator by Dreier [13]. For each instance tested, we report solutions for 3 values of k, the number of nodes to be deleted.

As a basis for comparison, we have implemented the integer programming model for the CRITICAL NODE PROBLEM using the CPLEX™ optimization suite from ILOG [12]. CPLEX contains an implementation of the simplex method [21], and uses a branch and bound algorithm [33] together with advanced cutting-plane techniques [22, 28].

We begin by providing the results from the terrorist network [24]. The graph, which is shown in Figure 8 has 62 nodes and 153 edges. Notice that 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 seconds of computation time. The average time to compute the optimal solution using CPLEX was 5387.31 seconds. Clearly even for this relatively small network, the heuristic is the method of choice. Figure 9 shows the resulting graph of the terrorist network according to the optimal solution to the CNP for the instance of $k = 20$.

In order to determine the 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 required by CPLEX, and finally the heuristic solution and the corresponding computation time. For each graph, we report solutions for 3 different values of k.

Notice that for all 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 seconds for all 27 instances. On the other hand,

Instance	IP Model		Heuristic		Heuristic + LS	
Nodes Deleted (k)	Objective Value	Execution Time (s)	Objective Value	Execution Time (s)	Objective Value	Execution Time (s)
20	20	12.69	22	0.08	20	0.01
15	61	277.77	66	0.03	61	0.01
10	169	3337.06	190	0.06	169	0.02
9	214	2792.33	229	0.15	214	0.02
8	282	15111.94	309	0.04	282	0.01
7	327	10792.08	329	0.09	327	0.01

TABLE 1. Results of IP model and heuristic on terrorist network data from [24].

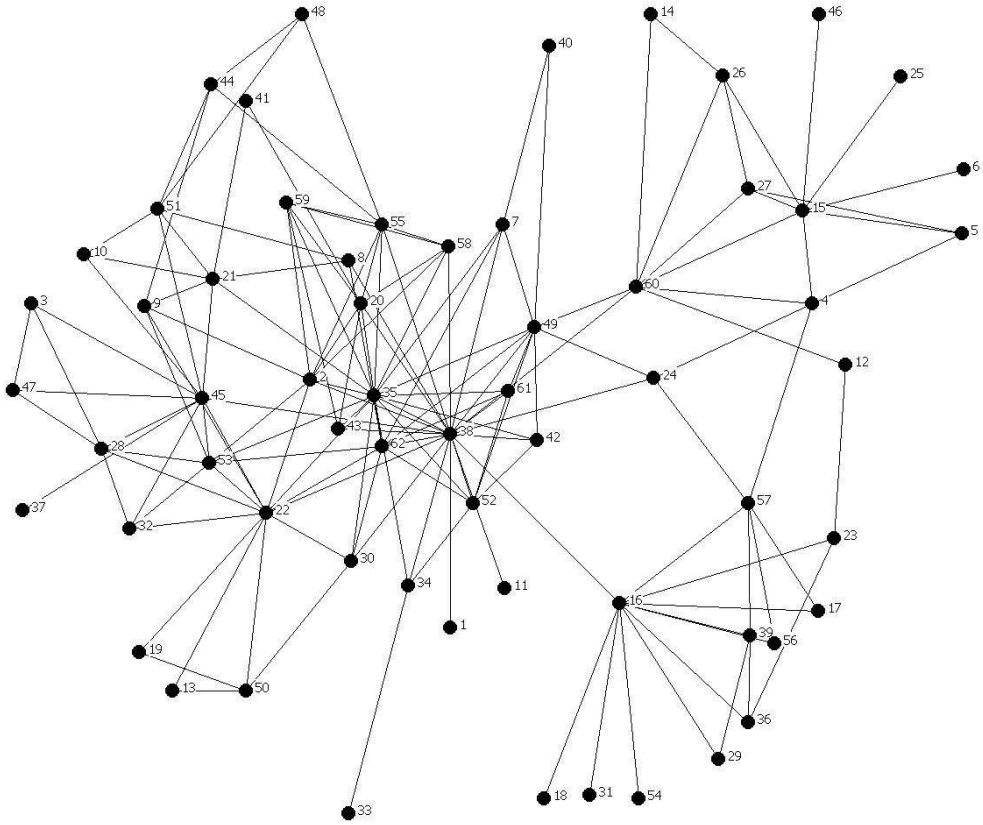

FIGURE 8. Terrorist network compiled by Krebs [24].

CPLEX required 289.44 seconds on average to compute the optimal solution, requiring over 5000 seconds 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.

4.2. **CC-CNP Results.** We continue with the results of the two algorithms developed for the CC-CNP, namely the combinatorial algorithm and the genetic algorithm. As above, we tested the IP model and both heuristics on the terrorist network [24] and a set of randomly generated graphs. For each instance tested, we report solutions for 3 values of L, the connectivity index threshold. Finally, we have implemented the integer programming model for the CC-CNP using CPLEX$^{\text{TM}}$.

Table 3 presents computational results of the IP model and heuristic solutions when tested on the terrorist network data. Notice that for all 5 values of L tested, the genetic algorithm and the combinatorial algorithm with local search (ComAlg + LS) computed optimal solutions. Figure 10 shows the optimal solution for the case when $L = 4$.

We now consider the performance of the algorithms when tested on the randomly generated data sets containing up to 50 nodes taken from [2]. The results are shown in Table 4. For these relatively small instances, we were able to compute the optimal solutions using CPLEX. For each instance, we provide solutions for 3 values of L, the maximum connectivity index. Notice that for these problems, the genetic algorithm computed optimal solutions for each instance tested in a fraction of the time required by CPLEX. The combinatorial heuristic found optimal solutions for all but 3 cases requiring approximately half of the time of the GA.

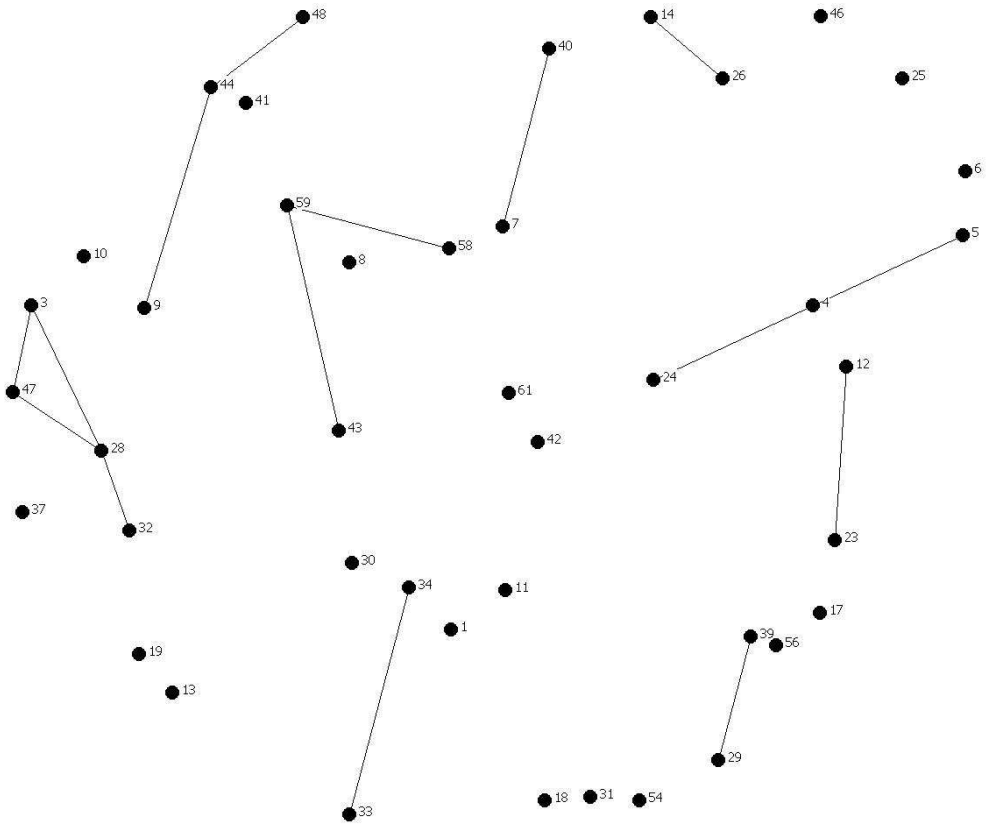

FIGURE 9. Optimal solution when $k = 20$.

Table 5 presents the solutions for the random instances from 75 to 150 nodes [2, 4]. Again, in order to demonstrate the robustness of the heuristics, we provide solutions for 3 values of L for each instance. In this table, we provide the results for the genetic algorithm and combinatorial heuristic with and without the local search enhancement. CPLEX was unable to compute optimal solutions within reasonable time limits for any of the instances represented in this table.

We see from this table that the in terms of solution quality the GA is the best performing method. The `ComAlg` + `LS` also favors well, but requires more computation time than the GA and requires more computing time on average. The combinatorial algorithm without the local search procedure produces solution which are arguably reasonable given that the required computation time is over 36 times faster than the GA, while the solutions are only 1.2 times worse than those computed by the GA. Nevertheless, the genetic algorithm required only 5.748 seconds on average to compute the best solution. The trade-off of solution quality versus computation time is a decision that would be made by an operator depending on the size of the network and the time constraints imposed on detecting the critical nodes of a given graph.

5. CONCLUDING REMARKS

In this chapter, we proposed several methods of jamming communication networks based on the detection of the critical nodes. Critical nodes are those vertices whose deletion results in the maximum network disconnectivity. In general, the problem of detecting

Instance			IP Model		Heuristic		Heuristic + LS	
Nodes	Arcs	Deleted Nodes (k)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)
75	140	20	36	66.7	92	0.12	36	0.03
75	140	25	18	33.28	39	0.28	18	0.03
75	140	30	7	4.23	18	0.02	7	0.04
75	210	25	26	93.71	78	0.1	26	0.04
75	210	30	8	3.57	31	0.05	8	0.05
75	210	35	2	4.36	16	0.18	2	0.04
75	280	33	26	749.19	54	0.00	26	0.04
75	280	35	20	164.34	38	0.09	20	0.06
75	280	37	13	83.98	24	0.39	13	0.11
100	194	25	44	151.14	142	0.731	44	0.09
100	194	30	20	59.66	72	0.56	20	0.11
100	194	35	10	8.51	33	0.66	10	0.12
100	285	40	23	136.47	48	1.151	23	0.11
100	285	42	17	263.82	38	0.4	17	0.17
100	285	45	11	16.78	29	0.53	11	0.23
100	380	45	22	128.13	58	0.58	22	0.15
100	380	47	16	243.07	42	1.191	16	0.16
100	380	50	10	228.72	23	0.31	10	0.11
125	240	33	62	5047.51	97	0.721	62	0.30
125	240	40	29	118.92	49	1.562	29	0.24
125	240	45	16	17.09	32	0.14	16	0.39
150	290	40	40	41.6	125	1.832	40	0.47
150	290	50	12	26.29	64	2.773	12	0.831
150	290	60	1	24.92	35	1.091	1	0.851
150	435	61	19	29.55	53	2.313	19	0.741
150	435	65	13	31.45	37	0.991	13	1.952
150	435	67	11	37.91	31	0.52	11	0.801

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

Instance	IP Model		Genetic Alg		ComAlg		ComAlg + LS	
Max Conn. Index (L)	Obj Val	Comp Time (s)	Obj Val	Comp Time (s)	Obj Val	Comp Time (s)	Obj Val	Comp Time (s)
3	21	188.98	21	0.25	22	0.01	21	0.1
4	17	886.09	17	0.741	19	0.01	17	0.45
5	15	30051.09	15	0.871	20	0.18	25	1.331
8	–	–	13	0.39	14	0.05	13	0.07
10	–	–	11	0.741	12	0.07	11	0.05

TABLE 3. Results of IP model and heuristics on terrorist network data from [24].

critical nodes has a wide variety of applications from jamming communication networks and other anti-terrorism applications, to epidemiology and transportation science [2, 4].

In particular we examined two problems, namely the CRITICAL NODE PROBLEM (CNP) as well as the CARDINALITY CONSTRAINED CNP (CC-CNP). Given a graph and an integer k, the objective of the CNP is to detect a set of k critical nodes whose deletion results in

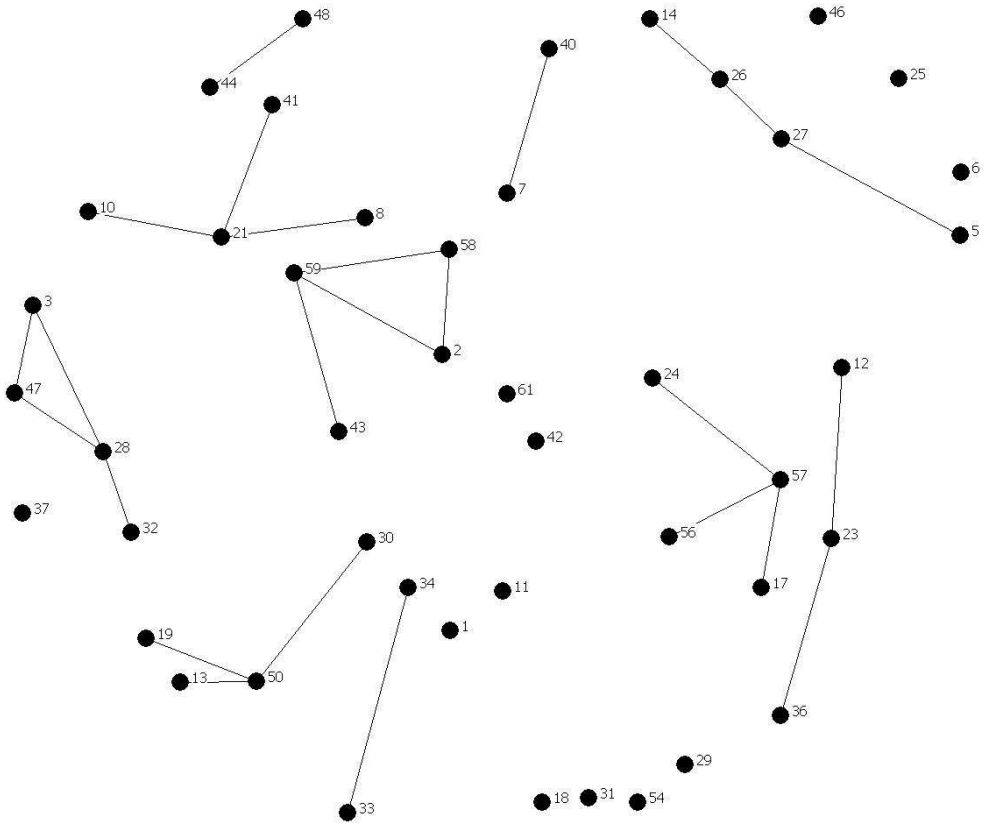

FIGURE 10. Optimal solution when $L = 4$.

the maximum number of disconnected components whose cardinalities have the minimum variance. The definition of the CC-CNP is slightly different in that instead of given $k \in \mathbb{Z}$, the maximum number of nodes to delete, we are given some value $L \in \mathbb{Z}$ which represents the maximum connectivity index a node may have. The objective in this case is to delete the minimum number of nodes while ensuring that the connectivity index of each node does not exceed L.

The proposed problems were modeled as integer linear programming problems. Then we proved that the corresponding decision problems are $\mathcal{NP}$-complete. Furthermore, we proposed a several heuristics for efficiently computing quality solutions to large-scale instances. The heuristic proposed for the CNP was a combinatorial algorithm which exploited properties of the graph in order to compute basic feasible solutions. The method was further intensified by the application of a local search mechanism. By using the integer programming formulation we were able to determine the precision of our heuristic by comparing their relative solutions and computation times for several networks. The computational experiments indicated that the heuristic found optimal solutions for all instances tested in a fraction of the time required by the commercial IP solver CPLEX.

For the CC-CNP we proposed two algorithms, namely a modified version of the combinatorial algorithm described above and a genetic algorithm [18]. Once again, the computational experiments indicated that both methods are robust and are able to efficiently compute approximate solutions for instances up to 150 nodes.

Instance			IP Model		Genetic Alg		ComAlg + LS	
Nodes	Arcs	Max Conn. Index (L)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)
20	45	2	9	0.04	9	0.02	9	0.03
20	45	4	6	0.13	6	0.04	6	0.862
20	45	8	5	0.39	5	0.04	5	1.482
25	60	2	11	0.07	11	0.49	11	0.08
25	60	4	9	14.1	9	2.113	10	0.01
25	60	8	7	26.64	7	0.05	8	0.06
30	50	2	11	0.07	11	0.06	11	0.01
30	50	4	8	0.1	8	0.05	8	0
30	50	8	6	1152.15	6	0.09	6	0
30	75	4	10	18.77	10	0.14	10	0.02
30	75	6	9	442.41	9	0.09	9	0.04
30	75	10	7	64.94	7	0.18	8	0
35	60	2	12	0.13	12	0.14	12	0.14
35	60	4	8	29.89	8	0.711	8	0
35	60	6	7	31.61	7	0.31	7	0.01
40	70	2	15	0.17	15	0.1	15	0.101
40	70	4	11	341.97	11	0.06	11	0
40	70	6	8	78.94	8	0.2	8	0.04
45	80	2	16	0.24	16	0.06	16	0.1
45	80	4	11	48.17	11	0.05	11	0.02
45	80	6	8	118.23	8	0.09	8	0.071
50	135	2	19	0.36	19	0.27	19	0.05
50	135	4	15	165.18	15	0.63	15	0.291
50	135	6	14	5722.88	14	0.721	14	0.03
Total (Sum)			24	8257.58	24	6.705	27	3.417

TABLE 4. Results of the IP model and genetic algorithm and the combinatorial heuristic on randomly generated scale free graphs.

We also conclude with a few words on the possibility of future expansion of this work. A heuristic exploration of cutting plane algorithms on the IP formulation would be an interesting alternative. Other heuristic approaches worthy of investigation include hybridizing the genetic algorithm with the addition of a local search or path-relinking enhancement procedure [17]. Finally, the local search used in the combinatorial algorithm was a simple 2-exchange method, which was the cause of a significant slow down in computation as noted in Table 5. A more sophisticated local search such as a modification of the one proposed by Resende and Werneck [29, 30] should be a major focus of attention.

Furthermore, it would be interesting to study the weighted version of the problem to see how weights added to the nodes affect the solutions. For example, it is rational to perceive applications containing weighted networks in which the cost of deleting one node is different from another. Also, pertaining to applications outside the scope of jamming networks, a study of epidemic threshold variation with respect to the heuristic results will help determine the impacts on contagion suppression in biological and social networks.

REFERENCES

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[2] A. Arulselvan, C.W. Commander, L. Elefteriadou, and P.M. Pardalos. Detecting critical nodes in social networks. *Social Networks*, submitted, 2007.

[3] A. Arulselvan, C.W. Commander, and P.M. Pardalos. A hybrid genetic algorithm for the target visitation problem. *Naval Research Logistics*, submitted, 2007.

Instance			Genetic Algorithm		ComAlg		ComAlg + LS	
Nodes	Arcs	Max Conn. Index (L)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)	Obj Value	Comp Time (s)
75	140	5	18	1.622	21	0	18	1.502
75	140	8	14	1.442	20	0.02	14	1.181
75	140	10	12	1.231	20	0.12	12	3.364
75	210	5	23	1.532	29	0.01	23	18.476
75	210	8	21	2.443	23	0.01	22	2.934
75	210	10	20	2.794	24	0.09	20	21.17
75	280	5	31	3.464	35	0.101	31	3.144
75	280	8	29	2.874	31	0.05	29	3.746
75	280	10	28	3.775	30	0.13	28	4.787
100	194	5	22	5.317	33	0.02	22	2.774
100	194	10	17	3.224	22	0.241	17	6.499
100	194	15	15	2.954	22	0.021	15	0.44
100	285	5	33	5.08	38	0.02	33	1.262
100	285	10	28	4.376	31	0.05	28	11.076
100	285	15	27	5.728	28	0.16	27	1.142
100	380	5	40	9.052	47	0.051	42	5.739
100	380	10	36	11.506	41	0.02	37	3.866
100	380	15	35	6.198	40	0.39	36	3.034
125	240	5	29	7.951	37	0.251	31	1.472
125	240	10	24	9.984	29	0.07	24	1.993
125	240	15	22	5.888	26	0.18	22	9.233
150	290	5	31	7.981	40	0.421	30	5.798
150	290	10	26	4.967	32	0.2	25	5.107
150	290	15	23	5.457	29	1.101	23	19.889
150	435	5	49	9.143	57	0.06	49	6.459
150	435	10	40	19.407	50	0.44	41	5.518
150	435	15	38	9.703	45	0.07	38	13.699
Total (Sum)			731	155.183	880	4.297	737	165.304

TABLE 5. Comparative results of the genetic algorithm and the combinatorial heuristic when tested on the larger random graphs. Due to the complexity, we were unable to compute the corresponding optimal solutions.

[4] A. Arulselvan, C.W. Commander, P.M. Pardalos, and O. Shylo. Managing network risk via critical node identification. In N. Gulpinar and B. Rustem, editors, *Risk Management in Telecommunication Networks*. Springer, submitted 2007.

[5] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statisical physics and circuit layout design. *Operations Research*, 36:493–513, 1998.

[6] A. Bavelas. A mathematical model for group structure. *Human Organizations*, 7:16–30, 1948.

[7] S.P. Borgatti. Identifying sets of key players in a network. *Computational, Mathematical and Organizational Theory*, 12(1):21–34, 2006.

[8] S.I. Butenko. *Maximum Independent Set and Related Problems, with Applications*. PhD thesis, University of Florida, 2003.

[9] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 85:4626, 2000.

[10] D.A. Coley. *An introduction to genetic algorithms for scientists and engineers*. World Scientific, 1999.

[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[12] ILOG CPLEX. http://www.ilog.com/products/cplex, Accessed October 2006.

[13] D. Dreier. Barabasi graph generator v1.4. http://www.cs.ucr.edu/ ddreier, Accessed November 2006.

[14] L. Elefteriadou. Highway capacity. In M. Kutz, editor, *Handbook of Transportation Engineering*, chapter 8, pages 8–1 – 8–17. McGraw-Hill, 2004.

[15] L.C. Freeman. Centrality in social networks I: Conceptual Clarification. *Social Networks*, 1:215–239, 1979.

[16] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
[17] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path-relinking. *Control Cybernetics*, 39:653–684, 2000.
[18] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
[19] R.E. Gomory and T.C. Hu. Multi-terminal network flows. *Journal of SIAM*, 9(4):551–570, 1961.
[20] P.R. Harper, V. de Senna, I.T. Vieira, and A.K. Shahani. A genetic algorithm for the project assignment problem. *Computers and Operations Research*, 32:1255–1265, 2005.
[21] F.S. Hillier and G.J. Lieberman. *Introduction to Operations Research*. McGraw Hill, 2001.
[22] R. Horst, P.M. Pardalos, and N.V. Thoai. *Introduction to Global Optimization*, volume 3 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, 1995.
[23] R.M. Karp. Reducability among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
[24] V. Krebs. Uncloaking terrorist networks. *First Monday*, 7(4), April 2002.
[25] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.
[26] H. Narayanan, S. Roy, and S. Patkar. Approximation algorithms for min-k-overlap problems using the principal lattice of partitions approach. volume 21, pages 306–330, 1996.
[27] C.A.S. Oliveira, P.M. Pardalos, and T.M. Querido. Integer formulations for the message scheduling problem on controller area networks. In D. Grundel, R. Murphey, and P. Pardalos, editors, *Theory and Algorithms for Cooperative Systems*, pages 353–365. World Scientific, 2004.
[28] C.A.S. Oliveira, P.M. Pardalos, and T.M. Querido. A combinatorial algorithm for message scheduling on controller area networks. *International Journal of Operations Research*, 1(1/2):160–171, 2005.
[29] M.G.C. Resende and R.F. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operations Research*, 174:54–68, 2006.
[30] M.G.C. Resende and R.F. Werneck. A fast swap-based local search procedure for location problems. *Annals of Operations Research*, published online, DOI: 10.1007/s10479-006-0154-0, 2007.
[31] W.M. Spears and K.A. DeJong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.
[32] J.M. Wislon. A genetic algorithm for the generalised assignment problem. *Journal of the Operational Reswearch Society*, 48:804–809, 1997.
[33] L. Wolsey. *Integer Programming*. Wiley, 1998.
[34] T. Zhou, Z-Q. Fu, and B-H. Wang, editors. *Epidemic dynamics on complex networks*. 452-457, Progress in Natural Science, 2006 16.

(A. ARULSELVAN) CENTER FOR APPLIED OPTIMIZATION, DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, GAINESVILLE, FL, USA.
E-mail address: ashwin@ufl.edu

(C.W. COMMANDER) AIR FORCE RESEARCH LABORATORY, MUNITIONS DIRECTORATE, AND, DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, GAINESVILLE, FL USA.
E-mail address: clayton.commander@eglin.af.mil

(P.M. PARDALOS) CENTER FOR APPLIED OPTIMIZATION, DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, GAINESVILLE, FL, USA.
E-mail address: pardalos@ufl.edu

(O.V. SHYLO) CENTER FOR APPLIED OPTIMIZATION, DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, GAINESVILLE, FL, USA.
E-mail address: shylo@ufl.edu