A mixed-integer programming approach for locating jamming devices in a flow-jamming attack

Satish Vadlamani[1], Hugh Medal[1,*], David Schweitzer[1], Apurba Nandi[1], Burak Eksioglu[2]

[1] *Department of Industrial and Systems Engineering, Mississippi State University P.O. Box 9542, Starkville, MS 39762.*

[2] *Department of Industrial Engineering Clemson University 272 Freeman Hall Clemson, SC 29634*

Abstract

The ubiquitous nature of wireless networks makes them increasingly prone to jamming attacks as such attacks become more sophisticated. In this paper, we seek to gain understanding about a particular type of jamming attack: the flow-jamming attack. Toward this end, we provide a mixed-integer programming model for optimizing the location of jamming devices for flow-jamming attacks. An accelerated Benders decomposition approach was used to solve the model. We solved the problem for two realistic networks and 12 randomly generated networks and found that the Benders approach was computationally faster than CPLEX for nearly all the problem instances, particularly for larger problems with 1000 binary variables. The experimental results show that optimally locating jamming devices can increase the impact of flow-jamming attacks. Specifically, as the number of possible locations increases the jammers' efficacy increases as well, but there is a clear point of diminishing returns. Also, adding lower-powered jammers to work in conjunction with higher powered jammers significantly increases overall efficacy in spite of the power difference.

Keywords: OR in telecommunications; OR in defense; flow-jamming attacks;jamming device placement problem; Benders decomposition

1. Introduction

The goal of this paper is to study the jammer placement problem for flow-jamming attacks. Toward this end, we develop a mixed-integer programming (MIP) model that maximizes the impact of jamming by maximizing the fraction of flows being jammed. We also present an accelerated Bender's decomposition algorithm and use the model and algorithm to gain insight into this particular jammer placement problem.

*Corresponding author

[1] Email : hugh.medal@msstate.edu,Tel.:+1(662)325-3923; fax:+1(662) 325-7618

1.1. Motivation

Wireless networks refer to the communication in networks between devices, such as laptops, without the use of cables or wires. Common types of wireless networks include wireless local area networks (WLANs), wireless sensor networks (WSNs), and ad hoc networks (AHNs). A WLAN, also known as a Wi-Fi network, allows for devices to easily connect to the Internet whenever needed as long as the device can connect to a wi-fi signal. Such networks are increasingly ubiquitous, and can be found in schools, homes, and coffee shops [1]. A WSN is a collection of a large number of autonomous nodes that gathers information from the area in which they are deployed and shares information among other nodes or sends the information to a base station [2]. An AHN is a temporary connection in which devices communicate with one another directly without the use of a router signal, and are typically used when building a well-established network infrastructure is not possible such as in a disaster or military situation [3].

Because wireless networks by definition rely on air as a medium for data transfer instead of cables or wires, they are highly susceptible to attacks. With increased dependence on networked information systems, finding better ways to secure them is increasingly important. Among the various attacks in wireless networks, Worm hole attacks [4], Sybil attacks [5], and jamming attacks cause the most security concern [6] because these attacks are easy to launch. Although not discussed within this paper, there are several other attacks and defensive strategies in wireless networks [7].

1.2. Related Literature

Wireless jamming attacks, a type of Denial of Service (DoS) attack [8], have been widely researched [9, 10] and are employed by the military to deny terrorists the ability to transfer data through a network, as an example. Data sent by a source node travels through different abstract layers before they reach the destination node [2, 11]. A DoS attack on the physical layer is called a jamming attack [11]. In jamming attacks, the jamming device transmits radio signals that disrupt communications in the network by decreasing the Signal-to-Interference-plus-Noise ratio (SINR) [12], which is the ratio of the signal power to the sum of the interference power from other interfering signals and noise power. A desirable ratio greater than 1 indicates more signal than noise, and with enough power and by choosing the same frequency as the network's frequency, coupled with the same type of modulation, the jamming device can override any signal in the network. Jamming devices transmit radio signals which could, in the presence of legitimate signal, disrupt communication. There are many types of jammers like, constant jammer, reactive jammer, random jammer, and deceptive jammer . Constant jammers continously emit radio signals or random sequence of bits that disrupt legitimate communication. Reactive jammers do not constantly jam and conserve power, but react to the presence of a legitimate signal and jam it. Random jammers like reactive jammers conserve

energy by not constantly jamming, and randomly switching between jamming and sleep states. Deceptive jammers like constant jammers continuously emit radio signals, but unlike constant jammers, decepetive jammers do not send random bits, but send legitimate bits giving a impression of a legitimate node to the network. For better understanding jammers and how jamming attacks work, refer to [7] and [13].

As well as the physical layer, Thuente and Acharya [14] and Wood and Stankovic [15] have also studied jamming attacks on the link layer. Tague et al. [16] introduced a more sophisticated jamming attack that uses higher layer information to jam the data flowing through the network. They studied the problem of intelligently assigning jamming devices to the flow of data in the network and referred to the attack as a *flow-jamming attack*. A single packet travels through multiple wireless links, and an adversary planning to jam a network has a limit on the amount of power available in the jamming device; hence, the adversary should choose to jam when minimal energy is required to jam. A smart attacker can disrupt communication significantly by using higher layer information (e.g., the network layer), less power, and by intelligently assigning jamming devices to the data flowing in the network.

Among the jamming literature there are a few papers that deal with location problems in jamming attacks. Wood et al. [17] created a mapping detection approach to provide feedback to the base station about further jamming areas and power management strategies for the nodes that are under jamming attack or within the range of the jamming devices. Liu et al. [18] addressed the problem of finding a jamming device located in a wireless network by proposing a least-squares-based localization algorithm that estimated the location of the jamming device by using the changes in neighboring nodes caused by the presence of a jamming device. Liu et al. [19] proposed a method to find the location of multiple jamming devices in a network even when the jamming areas overlap. Such studies contribute to finding the location of a jamming device as a defense strategy. For attacking strategies, determining the optimal number and/or the optimal placement of a set of jammers is also an area of research, a problem introduced by Commander et al. [20], known as the Wireless Network Jamming problem. The authors developed an integer programming model for finding the minimum number of jamming devices needed to meet a certain jamming threshold. Vadlamani et al. [21] solved a bi-level min-max jammer placement problem in which an attacker places jamming devices to minimize the throughput of the network, and the defender's objective is to maximize the throughput of the network by solving a max flow problem.

In addition, a number of researchers have studied the network interdiction problem, a problem that is closely related to the problem of jamming a wireless network. Researchers have studied interdiction problems for objectives such as minimizing the maximum flow [22], maximizing the shortest path [23], and minimizing network connectivity [24, 25]. Building upon this work, researchers have also considered networks with hub-and-spoke structure [26] and series-parallel networks [27]. Further, extensions have been made to consider

multiple commodities [28], multiple time periods [29], dynamic two-player interactions [30], and random interdiction effect [31]. Researchers are continuing to study network interdiction problems in a diverse set of areas such as facility location [32, 33, 34, 35, 36], disease control [37], and cyber security [38].

Tague et al. [16] were the first to introduce flow-jamming attacks in a wireless network and defined various evaluation metrics to measure the impact of such attacks; they assumed jamming device locations were known and solved a linear programming model to assign jamming devices in order to optimize the metrics. For flow-jamming attacks in multichannel wireless networks, Kim et al. [39] proposed stochastic search algorithms like iterative improvement, simulated annealing, and a genetic algorithm to provide a stochastic optimization approach. However, both of these papers assumed known locations of jamming devices in their study.

However, the work in papers Commander et al. [20] and Vadlamani et al. [21] cannot be directly applied to flow-jamming attacks because flow-jamming attacks use higher layer information, such as the network layer information, when planning how to attack. In addition, although Tague et al. [16] and Kim et al. [39] study flow-jamming attacks, they assumed that the locations of the jamming devices are known. In summary, the jammer location problem has not been studied for flow-jamming attacks. As a result, there is currently a lack of understanding about how different jammer placement decision impacts the throughput loss in flow-jamming attacks. Thus, there is a need to study the jammer location problem in this context.

1.3. Contributions

In this paper, we discuss the flow-jamming attack by looking at the problem from the attackers point of view and study the jammer placement problem. Specifically, this paper makes the following contributions:

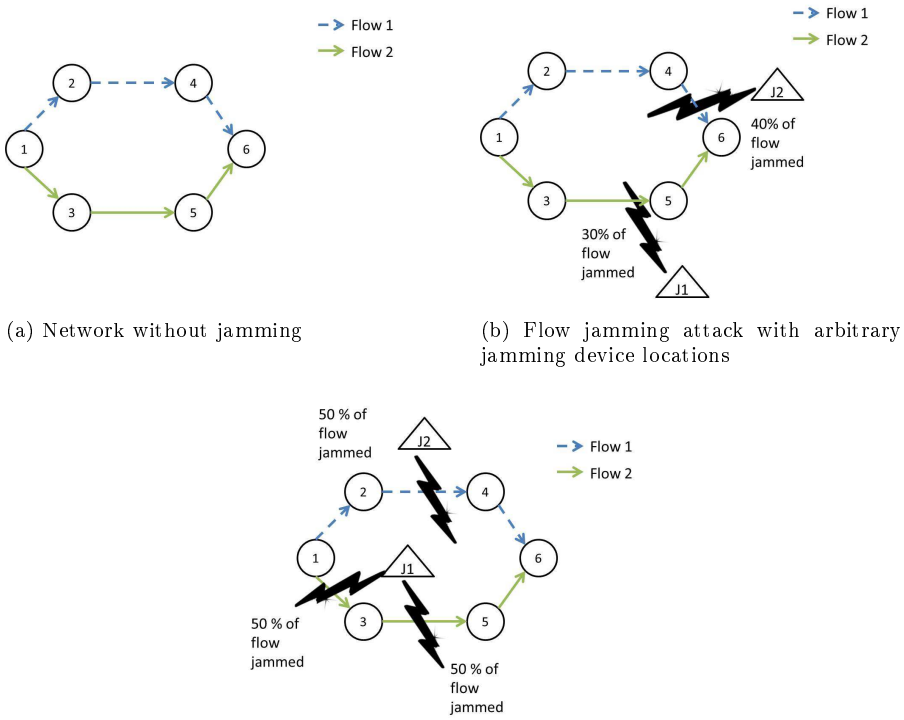
1. A mixed-integer programming model for analyzing the impact that jamming device locations have on jamming effectiveness,
2. An accelerated Benders decomposition algorithm, and
3. We analyze how the model responds to changes in parameter values; this analysis helps understand the impacts that changes to parameters have on jamming efficacy.

The rest of the paper is organized as follows: Section 2 describes the problem at hand and provides a mixed-integer programming model. Section 3 describes the Benders decomposition algorithm and the different acceleration techniques used. Section 4 provides the computational results and discussion of the results. Finally, Section 5 concludes the paper.

2. Problem Description and Mathematical Model

The main objective of this paper is to develop a mixed integer programming (MIP) model for flow-jamming attacks which takes into account the impact of the location of jamming devices on the flow. A flow in a network is a path on which the data or packets are sent from the source to the destination. The problem is approached from the attacker's point-of-view, and it consists of optimally placing a set of jamming devices onto a given set of possible locations such that the impact of the attack is maximized. It is to be noted that since we are studying the problem from an attackers point-of-view, we do not study the localization of jammer (see [18]) and concentrate only on the optimal jammer placement problem. The adversary can choose to jam each packet sent by the network when minimal energy is required because a single packet traverses multiple wireless network links, effectively jamming the traffic flow. The attacker, who has multiple jamming devices, aims to use minimal power from the total available power to jam network flows over multiple jamming devices and to maximize the amount of disruption. Hence, the efficiency of the attack can be optimized by intelligently assigning jamming devices to flows, which is especially important when considering the possibility that different jammers have different power levels. This can be thought of as a battlefield scenario where military strategists try to jam flows or packets of information from one terrorist camp to another. They ideally want to jam all the flow that is transmitted between the terrorist camps, but they have limited power levels. So, optimizing the jamming attack to get the maximum benefit is the goal of the military strategists. A key assumption in our model is that the path for each flow is fixed; that is, the network does not re-route traffic to avoid jamming. This assumption is appropriate because the low energy usage of jamming attacks makes them very difficult for the network operator to detect, in turn making re-routing difficult.

Figure 1 is an illustrates the effect of optimally locating jamming devices prior to a flow-jamming attack. This is not a real problem we solve in this paper, but an example of the importance of jammer placement problems in flow-jamming attacks. Figure 1a shows a network with six nodes and two flows; Flow 1 and Flow 2. The flows are not under a jamming attack, and hence, they can each transmit 100% of their data. However, if the network is hit by a flow-jamming attack, some of the data may be lost. Suppose that an attacker launches a flow-jamming attack and places the devices as shown in Figure 1b. In this case, 40% of Flow 1 and 30% of Flow 2 is jammed. If the jammer solves an optimization problem to find the optimal location of jamming devices, as shown in Figure 1c, 50% of Flow 1 and 100% of Flow 2 is jammed. Therefore, it is important for the military to find the optimal location of jamming devices with an aim to maximize the impact of flow-jamming attacks.



(a) Network without jamming

(b) Flow jamming attack with arbitrary jamming device locations

(c) Flow jamming attack with optimal jamming device locations

Figure 1: Example of flow-jamming attacks

In this paper, we model a wireless network as a graph whose nodes represent wireless transmitters and whose arc represent potential communications between transmitters. The set of nodes is denoted as $\mathcal{N}$. The data or packet flows between the source and destination nodes in $\mathcal{N}$ is considered to be a set of single path flows $\mathcal{F}$. We assume that the location of the nodes and the number of flows do not change for the duration of flow-jamming attacks. It is a common problem in wireless networks that some of the packets or data sent by some of the nodes collide with concurrent transmission resulting in the loss of packets. As in Tague et al. [16] we assume that only one flow is scheduled at any given time, i.e., there will be no concurrent flows and hence any loss of packets is due to jamming. In the problem the jammer has a set of jamming devices represented by the set $\mathcal{I}$; each device has a varying amount of limited power. The set $\mathcal{J}$ be the set of locations at which the jamming devices can be placed. The adversary tries to locate the jamming devices in a way that will reduce the energy consumption and yet increase the impact of the flow-jamming attacks. Let c_{jf} be the cost to jam flow f by a jamming device at location j. The cost c_{jf} is proportional to the inverse of the squared distance from location j to the closest non-source node on the flow f (Tague et al. [16]). If r_f is the rate of flow in the network, i.e., the rate at which the packets are sent from the source to the destination in the network and is measured in bits/time, then $c_{jf}r_f$ is the total energy required for a jamming device i to jam every packet in flow f from location j. All the notations used in this paper are given in Table 1. Since every

Table 1: List of mathematical notations

Notation	Explanation
$\mathcal{N}$	Set of wireless network nodes
$\mathcal{F}$	Set of network flows
$\mathcal{J}$	Set of locations
$\mathcal{I}$	Set of jamming devices
r_f	Flow rate of flow f, $\forall f \in \mathcal{F}$
c_{jf}	Cost to jam flow f by a jamming device at location j $\forall j \in \mathcal{J}$, $f \in \mathcal{F}$
p_i	Jamming resource supply for jamming device i, $\forall i \in \mathcal{I}$

jamming device need not jam every packet in the flow f, we define a decision variable $0 \leq x_{ijf} \leq 1$ as the fraction of flow f that jamming device i at location j jams. Finally, when placing a jamming device i at location j, its available power, p_i, aids in determining the optimality of such a choice of placement, and so y_{ij} is a decision variable that is 1 if jamming device i is placed at location j and 0 otherwise.

We develop a mixed integer programming model for flow-jamming attacks as shown below:

$$\textbf{[}\boldsymbol{MIFJ}\textbf{]} \max \quad \sum_i \sum_j \sum_f \quad x_{ijf}$$

$$\sum_{j\in\mathcal{J}} \sum_{f\in\mathcal{F}} c_{jf} r_f x_{ijf} \leq p_i \ \ \forall i \in \mathcal{I} \tag{1}$$

$$\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} x_{ijf} \leq 1 \ \ \forall f \in \mathcal{F} \tag{2}$$

$$x_{ijf} - y_{ij} \leq 0 \ \ \forall i \in \mathcal{I}, j \in \mathcal{J}, f \in \mathcal{F} \tag{3}$$

$$\sum_{j\in\mathcal{J}} y_{ij} \leq 1 \ \ \forall i \in \mathcal{I} \tag{4}$$

$$\sum_{i\in\mathcal{I}} y_{ij} \leq 1 \ \ \forall j \in \mathcal{J} \tag{5}$$

$$0 \leq x_{ijf} \leq 1 \ \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, f \in \mathcal{F} \tag{6}$$

$$y_{ij} \quad \textit{binary} \ \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \tag{7}$$

The objective function of **[*MIFJ*]** is to maximize the total fraction of jammed flows. Constraint (1) is the power constraint. The total resource or power expenditure on the left hand side should be less than or equal to the total available power p_i for jamming device i. Constraint (2) is the flow constraint, which ensures that the amount of flow jammed by the jamming devices assigned to a flow f must not exceed the amount of flow sent. Constraint (3) ensures that a flow cannot be jammed by device i at location j unless device i is located

at location j. Constraint (4) forces each jamming device to be deployed to at most one location. Constraint (5) forces each location to have at most one jamming device. Constraint (6) enforces non-negativity and upper bound on the fraction of flow jammed, and constraint (7) is the binary constraint.

3. Algorithmic Strategy

The structure of **[*MIFJ*]** is such that we can separate the problem into two smaller problems: one with only binary variables and another with continuous variables. Taking advantage of this problem structure, we applied a Benders decomposition method (Benders [40]), a well-known decomposition method to solve mixed integer linear programs. The basic idea behind this method is to decompose the original problem into an integer *master problem* and a linear programming *subproblem*. In this section, we first provide a basic Benders decomposition formulation to solve [***MIFJ***]. Then, we present several approaches to speed up the algorithm.

3.1. Benders Decomposition

The underlying Benders reformulation for model **[*MIFJ*]** is given below:

$$\max \quad \textbf{[}\boldsymbol{MIFJ\text{-}SUB}\textbf{]}(x \mid \hat{y}) \tag{8}$$

subject to constraints (1), (2), (3), (4), (5), (6), and (7), where **[*MIFJ-SUB*]** is the Benders decomposition *subproblem*. For given values of $\{y_{ij}\}_{i\in\mathcal{I},\, j\in\mathcal{J}}$ variables that satisfied the integrality constraint (7), the model reduces to the following *primal subproblem* involving only the continuous variables $\{x_{ijf}\}_{i\in\mathcal{I},\, j\in\mathcal{J},\, f\in\mathcal{F}}$.

$$\begin{aligned}
\textbf{[}\boldsymbol{MIFJ\text{-}SUB}\textbf{]}(\hat{y}) \quad \max \quad & \textstyle\sum_i \sum_j \sum_f \quad x_{ijf} && && (9)\\
p_i^{-1} \sum_{j\in\mathcal{J}} \sum_{f\in\mathcal{F}} c_{jf} r_f x_{ijf} \quad & \leq && 1 \ \ \forall i \in \mathcal{I} && (10)\\
\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} x_{ijf} \quad & \leq && 1 \ \ \forall f \in \mathcal{F} && (11)\\
x_{ijf} - \hat{y}_{ij} \quad & \leq && 0 \ \ \forall i \in \mathcal{I}, j \in \mathcal{J}, f \in \mathcal{F} && (12)\\
0 \quad \leq x_{ijf} \quad & \leq && 1 \ \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, f \in \mathcal{F} && (13)
\end{aligned}$$

Let $\gamma = \{\gamma_i \geq 0 \mid i \in \mathcal{I}\}$, $\mu = \{\mu_f \geq 0 \mid f \in \mathcal{F}\}$, $\delta = \{\delta_{ijf} \geq 0 \mid i \in \mathcal{I}, j \in \mathcal{J}, f \in \mathcal{F}\}$, and $\pi = \{\pi_{ijf} \geq 0 \mid i \in \mathcal{I}, j \in \mathcal{J}, f \in \mathcal{F}\}$ be the dual variables for the constraints (10), (11), (12), and (13)

respectively. The dual of the *primal subproblem*, which we called the *dual subproblem* [***MIFJ-SUB(D)***] is given below:

$$[\textbf{\textit{MIFJ-SUB(D)}}] \quad \min \quad \sum_i \gamma_i + \sum_f \mu_f + \sum_i \sum_j \sum_f \hat{y}_{ij}\delta_{ijf} + \sum_i \sum_j \sum_f \pi_{ijf} \tag{14}$$

$$p_i^{-1} c_{jf} r_f \gamma_i + \mu_f + \delta_{ijf} + \pi_{ijf} \geq 1 \ \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, f \in \mathcal{F} \tag{15}$$

$$\gamma_i, \mu_f, \delta_{ijf}, \pi_{ijf} \geq 0 \tag{16}$$

Let ∇ be the polyhedron defined by the constraints (15) and (16), and let P_∇ be the set of extreme points in the feasible region of [***MIFJ-SUB(D)***]. Introducing an extra variable θ, we formulate the underlying Benders *master problem* **[*MIFJ-MP*]** as below:

$$\textbf{[\textit{MIFJ-MP}]} \max \quad \theta \tag{17}$$

$$\sum_j y_{ij} \quad \leq \quad 1 \ \ \forall i \in \mathcal{I} \tag{18}$$

$$\sum_i y_{ij} \quad \leq \quad 1 \ \ \forall j \in \mathcal{J} \tag{19}$$

$$\theta \quad \leq \quad \sum_i \gamma_i + \sum_f \mu_f + \sum_i \sum_j \sum_f y_{ij}\delta_{ijf} + \sum_i \sum_j \sum_f \pi_{ijf} \ \forall (\gamma, \mu, \delta, \pi) \in P_\nabla \tag{20}$$

$$y_{ij} \quad \textit{binary} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \tag{21}$$

Constraint (20) is often referred to as the Benders *optimality cut*. θ is bound by the objective value of the *dual subproblem* [***MIFJ-SUB(D)***]. We note that the Benders subproblem is always feasible; hence, only optimality cuts are needed in the Benders master problem. The model **[*MIFJ-MP*]**, although equivalent to the **[*MIFJ*]**, had a large number of optimality constraints (20) and exhaustively finding them was not efficient. Instead, we iteratively generated a subset of cuts that were sufficient to identify an optimal solution. In each iteration, we solved **[*MIFJ-MP*]** by replacing the set P_∇ with the subset $P_\nabla^n \subseteq P_\nabla$ of extreme points available at each iteration $n = 0, 1, 2, \ldots$. By solving the relaxed **[*MIFJ-MP*]** problem with only a subset of all the constraints, we got an upper bound for the original master problem. The idea behind the standard Benders decomposition is described below.

The algorithm starts by solving the relaxed master problem **[*MIFJ-MP*]** which provides a valid upper bound to the original problem. We represent this upper bound as UB. The optimal solution of the relaxed **[*MIFJ-MP*]**, given by $[\mathbf{z_{MP}^n}]$, was used to set up the dual sub problem [***MIFJ-SUB(D)***]. The value of $\hat{y}_{ij}^n$ in each iteration was the optimal solution obtained from the LP relaxation **[*MIFJ-MP*]**. The optimal solution

of the variables of [***MIFJ-SUB(D)***], provided the *optimality cut* for the relaxed master **[*MIFJ-MP*]**. The optimal objective function value, given by [$\mathbf{z^n_D}$] provided a lower bound (LB) to the original problem at each iteration. If the gap between the upper bound and lower bound falls below a predefined threshold value, ϵ, the algorithm terminates; otherwise P^n_∇ was updated by adding an optimality cut constraint (20). Pseudo-code of the standard Benders decomposition algorithm is shown in Algorithm 1.

Algorithm 1 Standard Benders Decomposition

Initialize, $UB = \infty$, $LB = 0$, $n = 1$, ϵ, $P_\nabla = \emptyset$
while true do
  Solve **[*MIFJ-MP*]** for y_{ij} and [$\mathbf{z^n_{MP}}$]
  if [$\mathbf{z^n_{MP}}$] $< UB$ **then**
    $UB = [z^n_{MP}]$
  end if
  For fixed values $\hat{y}_{ij}$, solve [***MIFJ-SUB(D)***] to obtain values of $(\gamma_i, \mu_f, \delta_{ijf}, \pi_{ijf}) \in P_\nabla$ and [$\mathbf{z^n_D}$]
  Generate extra cuts
  if [$\mathbf{z^n_D}$] $> LB$ **then**
    $LB = [\mathbf{z^n_D}]$
  end if
  if $(UB - LB)/UB < \epsilon$ **then**
    break
  else
    $P^{n+1}_\nabla = P^n_\nabla \cup \{(\gamma_i, \mu_f, \delta_{ijf}, \pi_{ijf})\}$
  end if
  $n = n + 1$
end while

We discuss generating extra cuts in detail in Section *3.2.2*

3.2. Accelerating Standard Benders Decomposition

In our initial experimentation, the standard Benders decomposition could not solve the problems in a reasonable amount of time (results provided in Section 4). Hence, we developed acceleration techniques to solve the **[*MIFJ-MP*]** problem faster and speed up convergence of Benders decomposition. The following subsections describe the acceleration techniques developed.

3.2.1. Pareto Optimality Cut

Magnanti and Wong [41] showed that if the dual subproblem [**MIFJ-SUB(D)**] has multiple optimal solutions, there could be a number of alternatives for the optimality cut constraint (20). They proved that, for the purpose of generating stronger cuts, adding cutting planes that are not dominated by other optimality cuts could improve convergence of the Benders decomposition. We say that a cut generated from an extreme point (γ^1, μ^1, δ^1, π^1) dominates a cut generated from another extreme point (γ^2, μ^2, δ^2, π^2) if and only if

$$\sum_i \gamma_i^1 + \sum_f \mu_f^1 + \sum_i \sum_j \sum_f y_{ij}\delta_{ijf}^1 + \sum_i \sum_j \sum_f \pi_{ijf}^1 \quad \leq \tag{22}$$
$$\sum_i \gamma_i^2 + \sum_f \mu_f^2 + \sum_i \sum_j \sum_f y_{ij}\delta_{ijf}^2 + \sum_i \sum_j \sum_f \pi_{ijf}^2$$

with a strict inequality for at least one point $y_{ij} \in Y$. Using the concept of core points, Magnanti and Wong [41] formulated a problem that generated Pareto-optimality cuts. A core point is defined as a point in the relative interior of the convex hull of the feasible region, and can be used as a proxy for the optimal solution. Let Y^{LP} be the polyhedron defined by (18), (19), and $0 \leq y_{ij} \leq 1 \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J}$. Let y^0 be the candidate core point in the Y^{LP} found by solving the LP relaxation of the **[*MIFJ-MP*]**. Even though such a LP relaxation solution is not guaranteed to be an efficient core point, it is often in the neighborhood of the integer optima (Santoso et al. [42]). Furthermore, after some cuts are added to the master problem, the LP relaxation solution will typically be in the interior of the convex hull of Y, and hence satisfy the requirement of being a core point (Santoso et al. [42]). We solve the subproblem shown below to obtain the Pareto-optimal cuts.

[*MIFJ-SUB(PO)*] min $\sum_i \gamma_i + \sum_f \mu_f + \sum_i \sum_j \sum_f y_{ij}^0 \delta_{ijf} + \sum_i \sum_j \sum_f \pi_{ijf}$ (23)

$$c_i^{-1} c_{jf} r_f \gamma_i + \mu_f + \delta_{ijf} + \pi_{ijf} \geq 1 \ \ \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, f \in \mathcal{F} \tag{24}$$

$$\sum_i \gamma_i + \sum_f \mu_f + \sum_i \sum_j \sum_f \hat{y}\delta_{ijf} + \sum_i \sum_j \sum_f \pi_{ijf} = z(\hat{\gamma_i}, \hat{\mu}_f, \hat{\delta}_{ijf}, \hat{\pi}_{ijf}) \tag{25}$$

$$\gamma_i, \mu_f, \delta_{ijf}, \pi_{ijf} \geq 0 \tag{26}$$

where $z(\hat{\gamma_i}, \hat{\mu_f}, \hat{\delta}_{ijf}, \hat{\pi}_{ijf})$ is the optimal solution of the dual problem. Constraints (24) and (26) enforce the dual feasible region, and constraint (25) restricts feasible dual solutions to the set of alternative dual optima. The objective function (23) corresponded to minimizing the value of the cut at y^0.

3.2.2. Round-Off Strategies

In this section, we discuss a round-off strategy applied to the relaxed master problem. Although there are many varieties of such round-off strategies (e.g., Thanh et al. [43], Bansal et al. [44]), we used the strategy as discussed by Costa et al. [45]. The main idea behind this strategy is to use the fractional solution obtained by solving the linear programming relaxation of the master problem to obtain approximations of the integer solution. Specifically, once a fractional solution is obtained, we sequentially round it off one variable at a time to obtain a set of partially-rounded solutions. To implement this within our Benders decomposition

approach, at each iteration we first solve a continuous relaxation of the master problem, obtaining a fractional solution. Second, we round the fractional variable values one at a time, obtaining a set of partially-rounded solutions. Finally, for each partially-rounded solution, we input it to the subproblem, solve the subproblem, and add a new Benders cut to the master problem. This approach allows us to obtain a set of additional Benders cuts with only the smaller amount of computation time needed to solve a continuous relaxation of the master problem. Moreover, as stated earlier, the subproblem is feasible for any y solution.

We generate an optimality cut of type (20) for each of the fractional variables generated whose value was greater than 0.5. The algorithm stops when this criterion was met. Thus, the rounding off strategy algorithm is:

Algorithm 2 Rounding off strategy

while fractional variables generated have a value greater than 0.5 **do**
  $(i, j) = \arg\max \{y_{ij} \in Y | y_{ij} < 1\}$
  Set $y_{ij} = 1$
  Solve subproblem
  Update LB
  generate optimality cut (20) and add to master problem
end while

3.2.3. Knapsack Inequality

Initial experimentation showed that, with an available good lower bound from the Benders decomposition algorithm, adding a knapsack inequality (KI) to the Benders master problem along with the optimality cut constraint (20) had a significant impact on the solution quality. In this paper, the primal subproblem is a maximization problem and hence we obtain a lower bound in each iteration. We add the knapsack cut as shown below to the master problem. A variety of valid inequalities from the knapsack inequality can be derived by modern commercial solvers such as CPLEX which would speed up convergence of the Benders algorithm [42]. Let LB be the best known lower bound obtained so far from the Benders algorithm. Since the Benders decomposition algorithm ensures that $LB \leq \theta$, we may derive the following knapsack inequality and add it to the master problem in iteration $n + 1$:

$$LB^n - \sum_i \gamma_i - \sum_f \mu_f - \sum_i \sum_j \sum_f \pi_{ijf} \quad \leq \quad \sum_i \sum_j \sum_f y_{ij} \delta_{ijf} \tag{27}$$

3.2.4. Solution Elimination Constraint

Brown et al. [46] introduced the idea of adding *solution elimination constraints* (SECs) to the master problem. Brown et al. [47] used SECs to ensure convergence of decomposition algorithms. Similarly, Israeli and Wood [23] referred to SECs as "supervalid inequalities" and showed that adding these constraints speeds

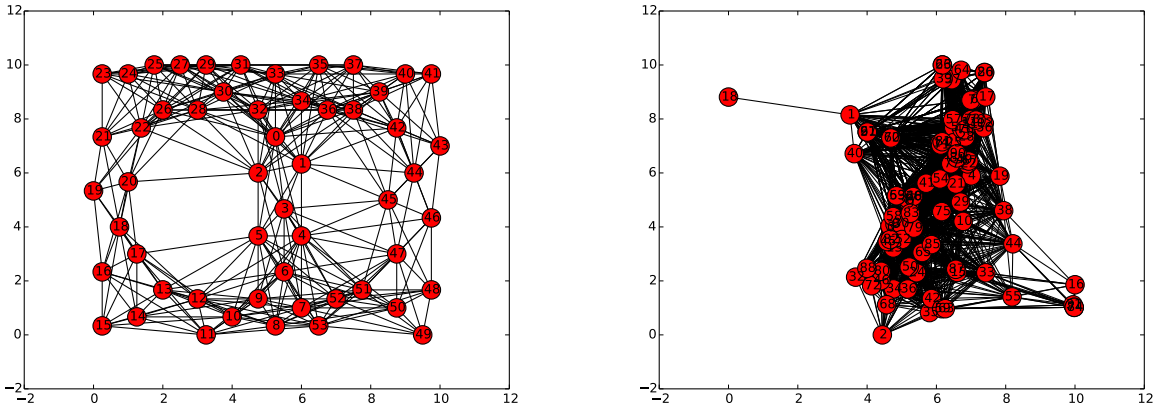
up convergence. We add the following SEC to our master problem:

$$\sum_{(i,j)|\mathcal{Y}^n=1} (1 - y_{ij}) + \sum_{(i,j)|\mathcal{Y}^n=0} y_{ij} \quad \geq \quad 1 \tag{28}$$

A cut (28) is added for every iteration n. The UB provided by solving the master problem **[*MIFJ-MP*]** with constraint (28) may not be valid if $\mathcal{Y}^n$ is optimal, and the bound may even drop below $[z^*]$. This occurs only if an optimal solution was at hand, and at this point the validity of the bound was irrelevant. Here $[\mathbf{z}^*]$ is the optimal solution of Algorithm 1 without the extra cuts. If **[*MIFJ-MP*]** is infeasible we set $UB = LB$. In this case the master problem **[*MIFJ-MP*]** is infeasible because all the solutions had been eliminated by SECs, and, thus, the solution $\mathcal{Y}^n$ obtained must be optimal.

4. Computational Results and Analysis

To test our approaches, we solved the flow-jamming attack problem on two different cases; the first was for two realistic networks CMU [48], MIT [49], and the second was on one set of randomly generated networks. The number of nodes was obtained from the realistic network's dataset. Figures 2a and 2b show the topology of the CMU and MIT networks, respectively. The randomly generated networks were created by uniformly generating node coordinates in a unit square. The set of flows $\mathcal{F}$ in the network were chosen by selecting pairs of origins and destinations at random from the network. The origin nodes were selected at random without replacement. In other words, we had a different origin node for each flow. In determining the intermediate nodes for each flow, the shortest path was found and used as the path for the flow. The number of flows in the network was also predefined, but the specific flows were determined using the aforementioned strategy for both the realistic and the random networks. All the experiments were run using the enhanced L-Shaped method (see Algorithm 1) implemented in Python 2.7 and run on a desktop computer with an Intel Core i7 2.70 GHz processor and 4.0 GB RAM. The optimization solver used was CPLEX 12.3. Below we describe each of the cases and provide the results.



(a) CMU (b) MIT

Figure 2: Network Topology of CMU and MIT

4.1. Experimental Results

Case 1:

In this case, we used the networks CMU and MIT, with 54 and 92 nodes, respectively. We assumed discrete locations for placing jamming devices. These discrete locations were constructed using 8×8 and 10×10 subgrids overlaid on a unit square, resulting in two levels of granularity. Figure 3 gives an example of a randomly-generated network with six nodes (1-6), two flows and 25 possible locations (shown as red stars) of the jamming devices overlaid on a network.

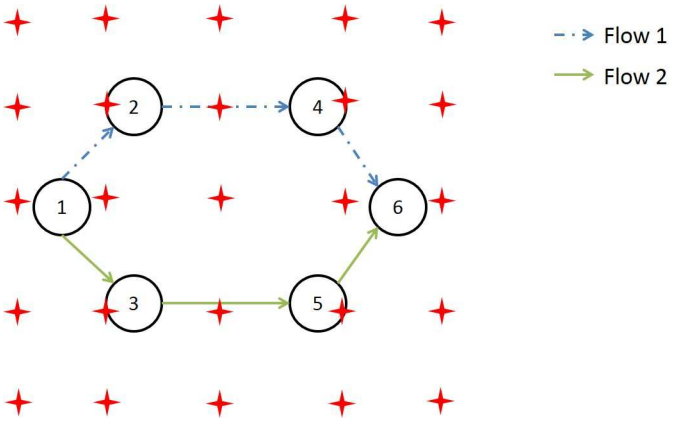


Figure 3: Example of Jamming Device Locations

It is apparent from Figure 3 that locations of the nodes and the locations of jamming devices were different. We had two types of jamming devices $T1$ and $T2$: the $T1$ jamming devices have a fixed power level of 1 mW, and $T2$ jamming devices have a fixed power level of 10 mW. The number of flows was fixed and the flows were decided using the strategy discussed above. Table 2 shows 12 different experimental problem instances for this case for both CMU and MIT.

Case 2:

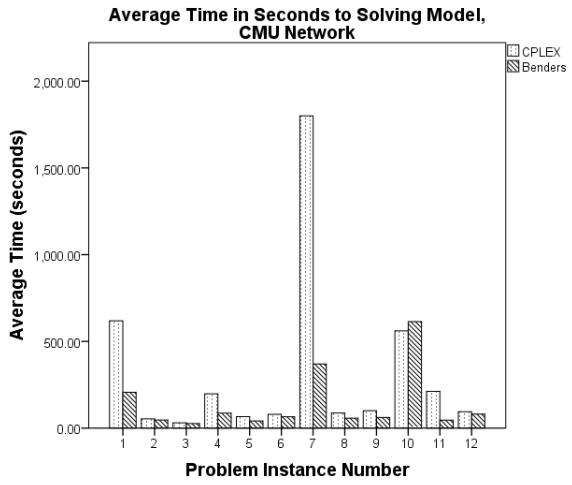
In this case, we solved a randomly generated 146 node network in a unit square. The links between these nodes were restricted by a predefined communication range. We used 10×10 possible discrete locations for the jamming devices, and the number of flows was 100. The flows were chosen using the strategy mentioned before. For each problem instance, 5 random networks were generated, and CPLEX and Benders were run on each of them to maintain consistency in comparing the two methodologies. However, for each problem instances 5 random networks were generated; thus, we used a total of 60 random networks.

Table 2: Problem Instances

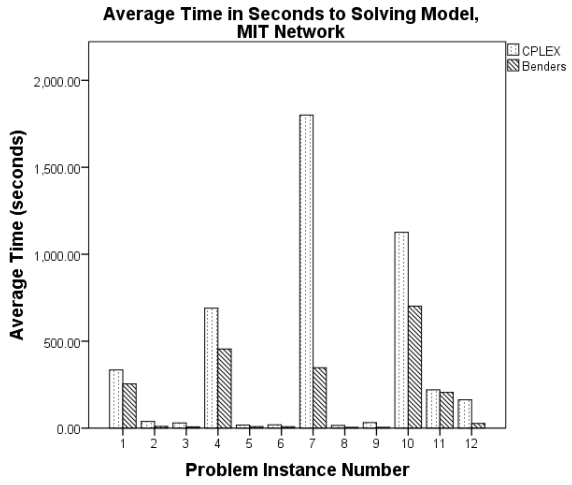
Instances	# Locations	T1	T2	# jamming devices	# Flows	Total Power
Problem 1	64	5	5	10	50	55
Problem 2	64	0	5	5	50	50
Problem 3	64	5	0	5	50	5
Problem 4	64	5	5	10	150	55
Problem 5	64	0	5	5	150	50
Problem 6	64	5	0	5	150	5
Problem 7	100	5	5	10	50	55
Problem 8	100	0	5	5	50	50
Problem 9	100	5	0	5	50	5
Problem 10	100	5	5	10	150	55
Problem 11	100	0	5	5	150	50
Problem 12	100	5	0	5	150	5

4.2. Results and Discussion:

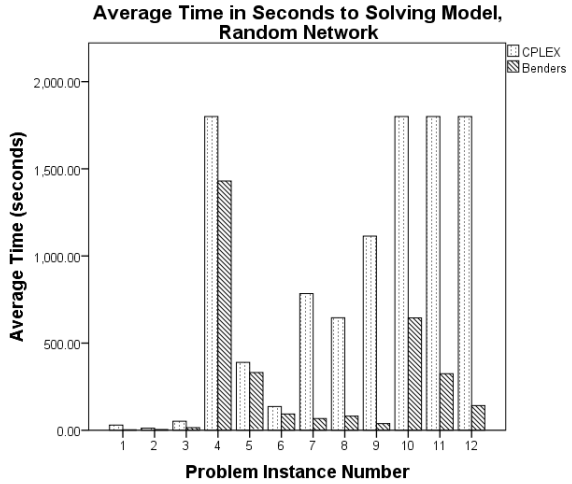
The computational results for the average objective value and average time to run for all three networks are shown in Figures 4 and 5, respectively. To account for noise associated with randomly generating networks, we report the average values over 5 trials.



(a) CMU



(b) MIT



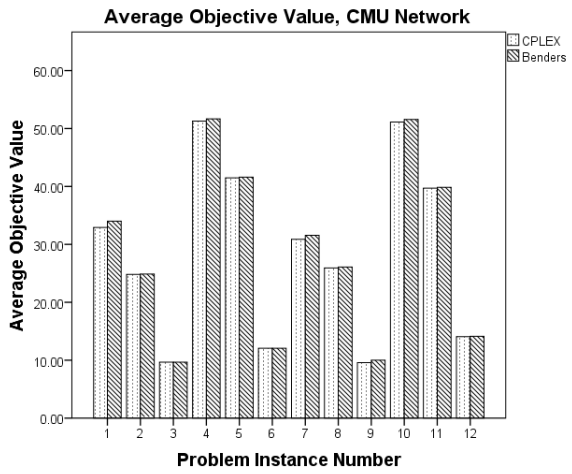
(c) Random

Figure 4: Average Computational Time (seconds) for Each of the Three Networks

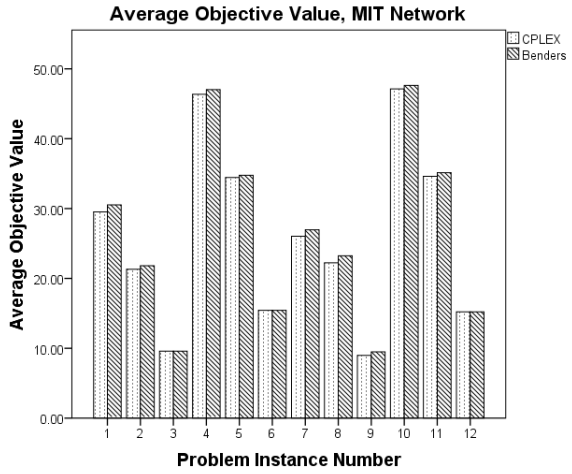
It is apparent from the results shown in Figure 4 that the Benders decomposition with all the acceleration techniques was faster than CPLEX for all of the problem instances for the networks with a single outlier, problem instance 10 in CMU.

From Figure 4, by comparing problem instances 1/7, 2/8, 3/9, 4/10, 5/11, and 6/12, one observation is that with the increase in the number of locations the computation time also increases for both the CMU and MIT networks. This analysis corresponded to grouping all the experimental runs where all the parameters are the same except for the number of potential jamming locations. There were, however, exceptions between problems 2 and 8 and 3 and 9 in MIT, where the 100 location problem was solved faster than the 64 location problem. These anomalies could have been attributed to the fact that the source and destination were

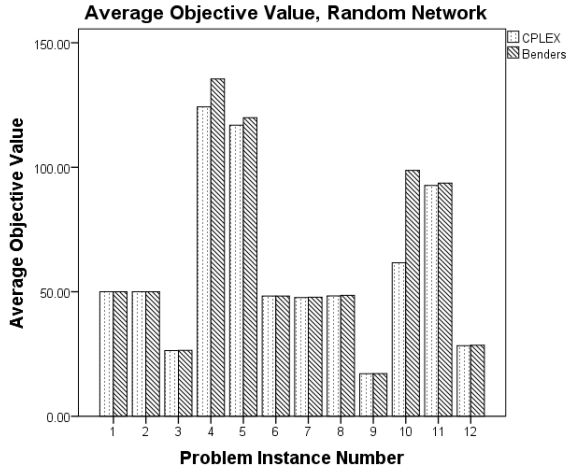
randomly chosen, but otherwise the computation time for CMU on the whole was higher than the MIT network; however, we did not know for certain the impact of the actual topologies of the networks. Another point of interest was that increasing the number of flows did not necessarily increase the computational time, nor did it necessarily decrease it. Certainly placing both $T1$ and $T2$ jammers required more time than placing either $T1$ or $T2$ jammers alone. However, by comparing problem instances 2/3, 5/6, 8/9, and 11/12, there was no clear indication that placing higher powered jammers took more or less time than lower powered jammers, which suggested the computational time of our model was independent of jamming power. As for how the jammer power affects jamming impact, we explore this idea more in Section 4.3.



(a) CMU



(b) MIT



(c) Random

Figure 5: Average Objective Value for Each of the Three Networks

The computational results for Case 1 shown in Figure 5 shows that as the number of flows increased the

total impact of jamming also increased. This is seen by comparing those experiments whose parameters were the same except the number of flows; in other words, problem instances 1/4, 2/5, 3/6, 7/10, 8/11, and 9/12. This was because of the fact that as the number of flows increases there were more flows in the network that could have been jammed. Another observation was that as the number of potential locations for jamming devices increased, the jamming devices had more options to locate a jamming device, thereby increasing the jamming impact.

Figure 5 also shows that with an increase in power the average impact of jamming increased. This was true for both CMU and MIT. So, providing the jamming devices with more power provided a significant increase in the impact of the jamming attacks. Since each of the problem instances presented were randomly generated (random topology for the Random network, random source and destination choices for all three networks) and had different run times, Figure 5 shows that the jamming impact in the flow-jamming attacks depended not only the number of locations, number of flows, number of jamming devices, and amount of power available, but also possibly on the topology of the network. We explore several of these parameters further in Section 4.3.

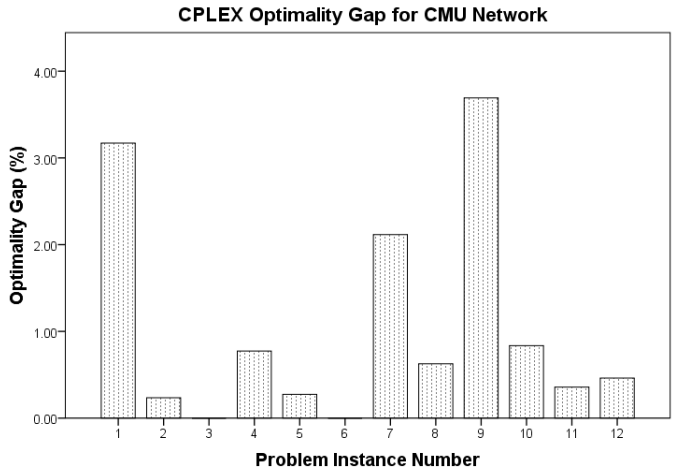
Now, in comparing our Benders algorithm to CPLEX directly, we discuss some observations of the results shown in Table 3. For each problem and for each of the three graphs, Table 3 shows the number of trials (out of 5) that CPLEX failed to find an optimal solution. We note that the enhanced Benders algorithm found the optimal solution within the 30 minute time limit in all of the problem instances.

Table 3: Instances where CPLEX does not optimally solve the model

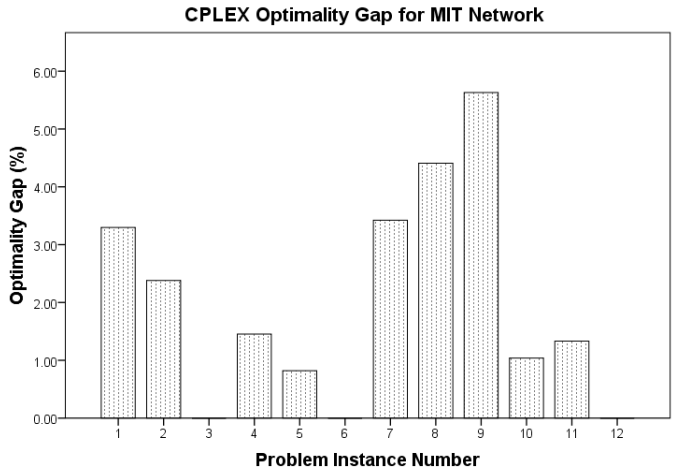
	1800 Second Limit Reached			Out of Memory			Optimality Gap Produced		
Problem Instance	CMU	MIT	Random	CMU	MIT	Random	CMU	MIT	Random
1	1	0	0	0	1	0	4	4	0
2	0	0	0	0	0	0	2	4	0
3	0	0	0	0	0	0	0	0	2
4	0	0	4	1	0	0	3	5	0
5	0	0	0	0	0	0	3	2	4
6	0	0	0	0	0	0	0	0	2
7	5	5	2	0	0	0	0	0	0
8	0	0	1	0	0	0	1	3	4
9	0	0	2	0	0	0	2	2	2
10	1	0	5	1	1	0	1	3	0
11	0	0	5	0	0	0	2	3	0
12	0	0	5	0	0	0	1	0	0

The 1800 Second Limit Reached group shows the number of trials during which CPLEX reached the 1800 second time limit, producing a feasible but not optimal solution. This specific issue happened most frequently on the Random network and in all cases happened in problem 7. The second major group shows how often CPLEX ran out of memory but still produced a feasible solution. The last major group shows

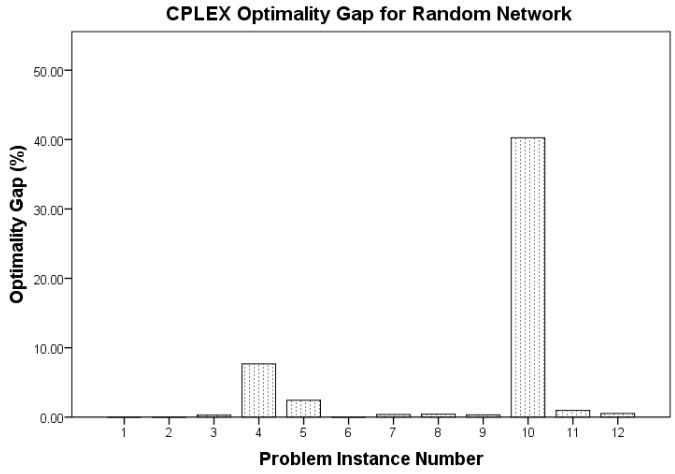
how often CPLEX could not optimally solve the problem, giving an optimality gap. We include the specific optimality gaps in Figure 6



(a) CMU



(b) MIT



(c) Random

Figure 6: Average Optimality Gap over 5 Trials

We remark that over all the instances, for the smaller problems with only 50 flows, it was unnecessary to use all our cuts to achieve a speed advantage over CPLEX, and so these were run with only the Pareto optimal cuts. However, for the larger problems with 150 flows, we employed all our cuts.

We also include the average number of iterations Benders took for each problem instance; we took the average over five trials because of the random choice of source and destination nodes. We show the results in Table 4. There was no obvious correlation between the parameter configurations and the number of iterations Benders needed to reach an optimal solution. However, with the exception of one problem instance on the

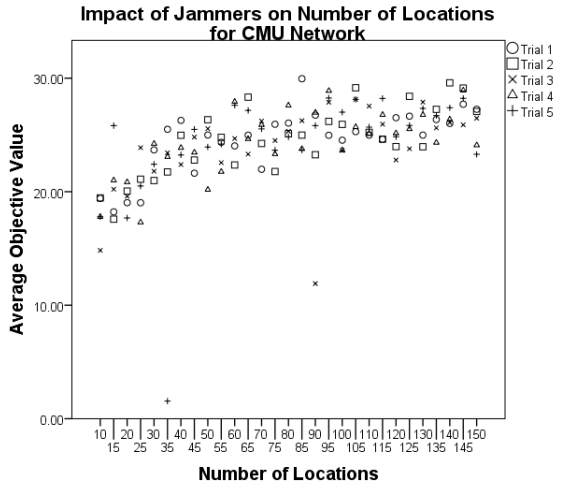
Random network, we needed fewer than 100 iterations overall. This result shows that our cuts work well both in allowing Benders to reach an optimal solution and in accelerating the algorithm since more iterations required more computational time.

Table 4: Average Number of Iterations for All Three Networks over 5 Trials

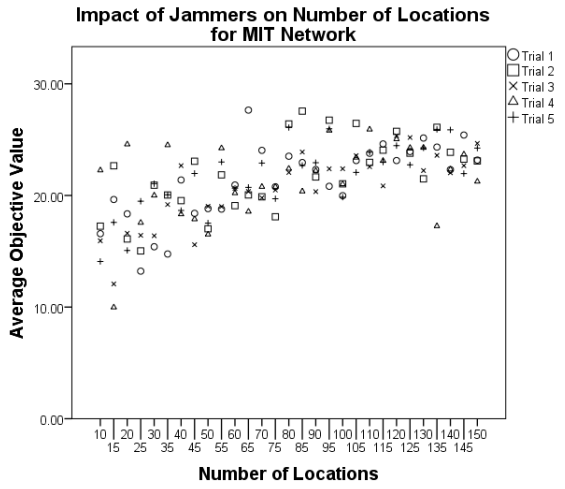
Problem Instance	CMU	MIT	Random
1	82.2	91.6	1
2	33.6	23.8	2.8
3	20.4	21.6	36.8
4	53.8	58.8	158.6
5	29.8	27.8	79.4
6	28.6	15	34.4
7	81.4	78.8	15.6
8	40.6	10.6	36
9	31.2	10.2	18.2
10	76.2	83.6	50.8
11	33.3	40.2	42.6
12	25.4	17.2	23

4.3. Sensitivity Analysis

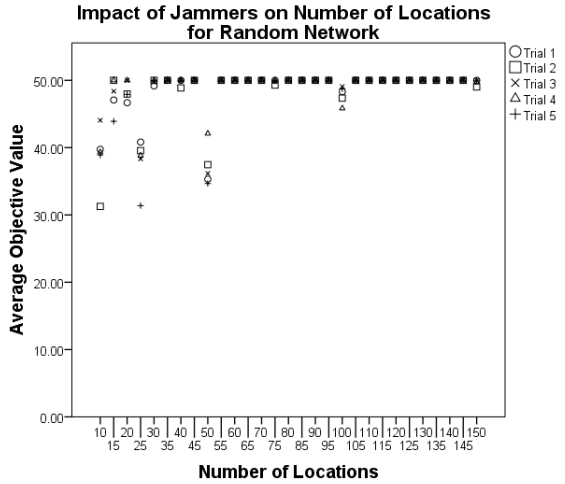
We now explore the impact that the number of jamming locations and additional devices with different power levels have on jamming efficacy. One question that this research seeks to answer is how great of an effect does increasing the number of potential locations for jammers have on the jamming impact? More precisely, is there a point of diminishing return? We provide the results of an experiment where, keeping the other parameters the same, we began with 10 possible locations for jammers and increased the amount by 5 until we reach 150. The results are shown in Figure 7.



(a) CMU network



(b) MIT network



(c) Random network

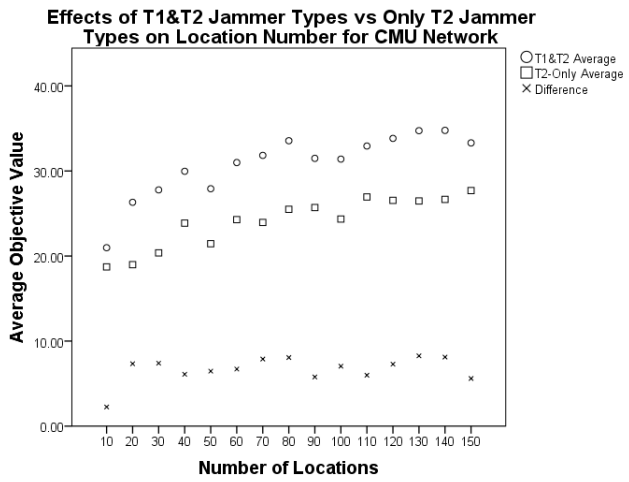
Figure 7: The effects of number of locations on jamming impact

For the CMU network, there was a clear point of diminishing return at half of our 150 possible locations; anything larger than 75 did not produce a significant gain in jamming efficacy. For the random network, this was true almost immediately as the jammers were readily able to jam nearly all the signal, though we provide a caveat that the random topology likely provided these results. For the MIT network, there was no clear contribution to increased jamming efficacy. These results suggest that a very coarse selection of suitable jammer locations was all that was necessary to attain a significant impact, and anything finer was both insignificant and more computationally expensive to determine.

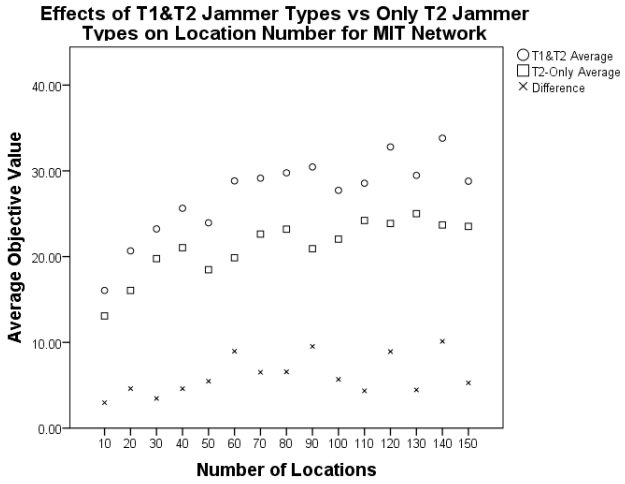
Next, as the $T1$ devices had a significantly lower power than the $T2$ devices, we wanted to determine the significance of using both sets of jammers in place of just the higher powered ones. Of course using both will

work better than using any one alone, but how much so was the key to our next experiment.

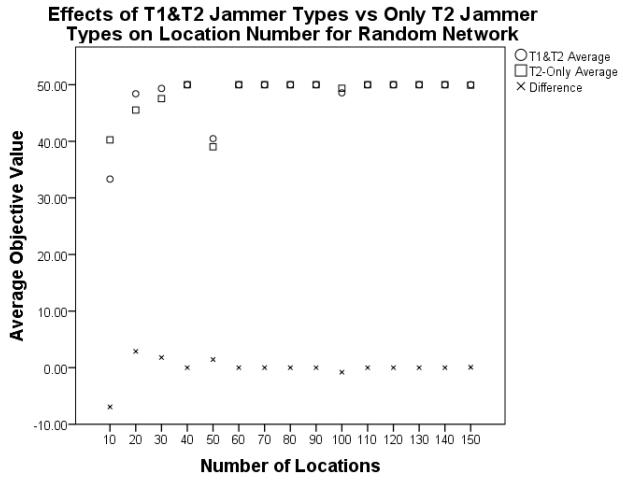
More jammers added to a network clearly increased jamming impact. But in the presence of high powered jammers, did the attacker need additional high powered jammers, or were additional low powered jammers able to add a significant effect to jamming efficacy? Our next experiment answered this question by comparing the $T2$ jammers' jamming impact to $T1$ and $T2$ jammers' impact, where the $T1$ jammer had only 10% of the $T2$ jammer's power. Figure 8 shows the results of this experiment.



(a) CMU network



(b) MIT network



(c) Random network

Figure 8: The effects of using both T1 and T2 devices over only T2 devices on the number of locations

Even though the weaker jammers only added 10% extra total power, there was nearly a 50% increase in jamming efficacy; thus, there was definitely an advantage to having the extra jammers as seen in the CMU and MIT networks. For the random network, however, there was no real significant advantage, but this was

likely the result of the randomness of the network itself where, from the previous results, nearly all the flow was jammed immediately.

Figure 8 also shows that as the number of jamming devices increased, the impact of a flow-jamming attack also increased. This was because the adversary had more jamming devices to place, which collectively increased the impact. Notice also that increasing the power level by switching $T1$ to $T2$ or using both $T1$ and $T2$ jammer types increased the impact of jamming on the flows in the network more than using $T1$ alone. With the higher amount of power available, the jamming devices jammed more flows, thereby increasing the impact of flow-jamming attacks.

5. Conclusion

This paper studies the jamming device placement problem for flow-jamming attacks on wireless networks. An MIP formulation for the flow-jamming attacks was developed to determine the optimal jamming device location in order to maximize the impact of flow-jamming attacks.

We found that there were instances where CPLEX took longer than the upper time limit of 30 minutes to solve the problem. To address this issue, we used a standard Benders decomposition algorithm to solve the problem. Due to the slow convergence of standard Benders decomposition algorithm, acceleration techniques to speed up the convergence are provided. Computational results show that the accelerated Benders decomposition algorithm can be used to solve realistic instances of large problems in reasonable time, in fact performing significantly faster than CPLEX for all problem instances except one. Using two realistic networks, CMU and MIT, and 12 random network instances, computational experiments were conducted to test the model and draw useful insights.

Observing the experimental results, it is apparent that different parameters, such as the number of jamming device locations, number of jamming devices, number of flows, and the amount of power available at each jamming device, play a very important role in the flow-jamming attacks. As seen, even a slight increase in total power through the addition of more jammers can greatly increase the impact of jamming, and while the number of their possible locations are important, this number does reach a point of diminishing returns. However, our model's computational time does not necessarily depend on jamming power alone; instead, as expected, our model is more dependent on the number of jammers when considering computational time. Finally, we have also shown that as the number of flows increases, the impact of jamming also increases.

5.1. Future Research

For future research, a detailed study on finding network topologies that are vulnerable to jamming attacks should be considered. The MIT and CMU networks used in this study had very specific topologies that may

or may not indicate a particular strategy in attacking or defending the network. Since this paper concentrates on the attacker's perceptive a mathematical programming model that incorporates the defenders strategies along with the attacker's strategies should also be developed for the flow-jamming attacks. The defender's strategies would minimize the impact of flow-jamming attacks, and a strategy to achieve this goal could be re-routing the packets through the network. This problem could be formulated as a bi-level attacker-defender problem, as done similarly in Vadlamani et al. [21].

References

[1] J. Geier, The state of wireless LANs, Tech. Rep., Network World Inc., 2004.

[2] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, G. Pantziou, A survey on jamming attacks and countermeasures in WSNs, Communications Surveys & Tutorials, IEEE 11 (4) (2009) 42–56.

[3] E. M. Royer, C. Toh, A review of current routing protocols for ad hoc mobile wireless networks, Personal Communications 6 (2) (1999) 46–55.

[4] Y. Hu, A. Perrig, D. B. Johnson, Packet leashes: a defense against wormhole attacks in wireless networks, in: INFOCOM, vol. 3, IEEE, 1976–1986, 2003.

[5] Q. Zhang, P. Wang, D. S. Reeves, P. Ning, Defending against Sybil attacks in sensor networks, in: Distributed Computing Systems Workshops, IEEE, 185–191, 2005.

[6] I. Shin, Y. Shen, Y. Xuan, M. T. Thai, T. Znati, A novel approach against reactive jamming attacks., Ad Hoc & Sensor Wireless Networks 12 (1-2) (2011) 125–149.

[7] S. Vadlamani, H. Medal, B. Eksioglu, Security in wireless networks: a tutorial, vol. 37 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, IOS Press, 272–288, 2014.

[8] K. Pelechrinis, M. Iliofotou, S. V. Krishnamurthy, Denial of service attacks in wireless networks: the case of jammers, Communications Surveys & Tutorials, IEEE 13 (2) (2011) 245–257.

[9] L. Lazos, S. Liu, M. Krunz, Mitigating control-channel jamming attacks in multi-channel ad hoc networks, in: Proceedings of the second ACM conference on Wireless network security, WiSec '09, ACM, New York, NY, USA, 169–180, 2009.

[10] Y. W. Law, M. Palaniswami, L. Van Hoesel, J. Doumen, P. Hartel, P. Havinga, Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols, ACM Trans. Sen. Netw. 5 (1).

[11] K. Bicakci, B. Tavli, Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks, Computer Standards & Interfaces 31 (5) (2009) 931–941.

[12] J. Berg, Broadcasting on the Short Waves, 1945 to Today, McFarland, Incorporated Publishers, 2008.

[13] S. Vadlamani, B. Eksioglu, H. Medal, A. Nandi, Jamming attacks on wireless networks: A taxonomic survey, International Journal of Production Economics 172 (2016) 76–94.

[14] D. Thuente, M. Acharya, Intelligent jamming in wireless networks with applications to 802.11 b and other networks, in: Proc. of MILCOM, vol. 6, 1–7, 2006.

[15] A. Wood, J. A. Stankovic, Denial of service in sensor networks, Computer 35 (10) (2002) 54–62.

[16] P. Tague, D. Slater, R. Poovendran, G. Noubir, Linear programming models for jamming attacks on network traffic flows, in: Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on, IEEE, 207–216, 2008.

[17] A. Wood, J. A. Stankovic, S. H. Son, JAM: a jammed-area mapping service for sensor networks, in: Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE, IEEE, 286–297, 2003.

[18] Z. Liu, H. Liu, W. Xu, Y. Chen, Exploiting jamming-caused neighbor changes for jammer localization, Parallel and Distributed Systems, IEEE Transactions on 23 (3) (2012) 547–555.

[19] H. Liu, Z. Liu, Y. Chen, W. Xu, Localizing multiple jamming attackers in wireless networks, in: Distributed Computing Systems (ICDCS), 2011 31st International Conference on, IEEE, 517–528, 2011.

[20] C. Commander, P. Pardalos, V. Ryabchenko, S. Uryasev, G. Zrazhevsky, The wireless network jamming problem, Journal of Combinatorial Optimization 14 (4) (2007) 481–498.

[21] S. Vadlamani, H. Medal, B. Eksioglu, P. Li, A Bi-Level Programming Model for the Wireless Network Jamming Placement Problem, in: E. by Y. Guan, H. Liao. (Eds.), Proceedings of the 2014 Industrial and Systems Engineering Research Conference, Montreal, Canada., 2014.

[22] Wood, Deterministic network interdiction, Mathematical and Computer Modelling 17 (2) (1993) 1–18, ISSN 08957177, doi:10.1016/0895-7177(93)90236-r.

[23] E. Israeli, R. K. Wood, Shortest-path network interdiction, Networks 40 (2) (2002) 97–111.

[24] A. Arulselvan, C. W. Commander, L. Elefteriadou, P. M. Pardalos, Detecting critical nodes in sparse graphs, Computers & Operations Research 36 (7) (2009) 2193–2200.

[25] D. Granata, G. Steeger, S. Rebennack, Network interdiction via a Critical Disruption Path: Branch-and-Price algorithms, Computers & Operations Research 40 (11) (2013) 2689 – 2702.

[26] T. L. Lei, Identifying Critical Facilities in Hub-and-Spoke Networks: A Hub Interdiction Median Problem, Geographical Analysis 45 (2) (2013) 105–122.

[27] S. Shen, J. C. Smith, Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs, Networks 60 (2) (2012) 103–119.

[28] C. Lim, J. C. Smith, Algorithms for discrete and continuous multicommodity flow network interdiction problems, IIE Transactions 39 (1) (2007) 15–26.

[29] A. Malaviya, C. Rainwater, T. Sharkey, Multi-period network interdiction problems with applications to city-level drug enforcement, IIE Transactions 44 (5) (2012) 368–380.

[30] B. J. Lunday, H. D. Sherali, A dynamic network interdiction problem, Informatica 21 (4) (2010) 553–574.

[31] K. J. Cormican, D. P. Morton, R. K. Wood, Stochastic network interdiction, Operations Research 46 (2) (1998) 184–197.

[32] F. Liberatore, M. P. Scaparra, M. S. Daskin, Analysis of facility protection strategies against an uncertain number of attacks: The stochastic R-interdiction median problem with fortification, Computers & Operations Research 38 (1) (2011) 357 – 366, project Management and Scheduling.

[33] A bilevel fixed charge location model for facilities under imminent attack, Computers & Operations Research 39 (7) (2012) 1364 – 1381.

[34] Y. Zhu, Z. Zheng, X. Zhang, K. Cai, The r-interdiction median problem with probabilistic protection and its solution algorithm, Computers & Operations Research 40 (1) (2013) 451 – 462.

[35] A bilevel partial interdiction problem with capacitated facilities and demand outsourcing, Computers & Operations Research 41 (2014) 346 – 358.

[36] N. Aliakbarian, F. Dehghanian, M. Salari, A bi-level programming model for protection of hierarchical facilities under imminent attacks, Computers & Operations Research 64 (2015) 210 – 224.

[37] A. K. Nandi, H. R. Medal, Methods for removing links in a network to minimize the spread of infections, Computers & Operations Research 69 (2016) 10 – 24.

[38] A. K. Nandi, H. R. Medal, S. Vadlamani, Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender-attacker model, Computers & Operations Research 75 (2016) 118 – 131.

[39] Y. S. Kim, B. DeBruhl, P. Tague, Stochastic Optimization of Flow-Jamming Attacks in Multichannel Wireless Networks, in: IEEE International Conference on Communications (ICC), 2165–2170, 2013.

[40] J. F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische Mathematik 4 (1) (1962) 238–252.

[41] T. L. Magnanti, R. T. Wong, Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria, Operations Research 29 (3) (1981) 464–484.

[42] T. Santoso, S. Ahmed, M. Goetschalckx, A. Shapiro, A stochastic programming approach for supply chain network design under uncertainty, European Journal of Operational Research 167 (1) (2005) 96–115.

[43] P. N. Thanh, N. Bostel, O. Peton, A DC programming heuristic applied to the logistics network design problem, International Journal of Production Economics 135 (1) (2012) 94 – 105.

[44] N. Bansal, A. Gupta, J. Li, J. Mestre, V. Nagarajan, A. Rudra, When LP is the Cure for Your Matching Woes: Improved Bounds for Stochastic Matchings, 2010.

[45] A. M. Costa, J.-F. Cordeau, B. Gendron, G. Laporte, Accelerating benders decomposition with heuristic master problem solutions, Pesquisa Operacional 32 (1) (2012) 03–20.

[46] G. G. Brown, R. F. Dell, R. K. Wood, Optimization and Persistence, Interfaces 27 (5) (1997) 15–37.

[47] G. G. Brown, W. M. Carlyle, R. C. Harney, E. M. Skroch, R. K. Wood, Interdicting a Nuclear-Weapons Project, Operations Research 57 (4) (2009) 866–877.

[48] CMU, Intel Lab Data, Internet, retrieved September 15, 2014, from http://www.select.cs.cmu.edu/data/labapp3/index.html, 2004.

[49] MIT, MIT roofnet nodes, Internet, retrieved September 15, 2014, from http://moment.cs.ucsb.edu/meshnet/datasets/, 2004.