Restoring Infrastructure Systems: An Integrated Network Design and Scheduling Problem

Sarah G. Nurre[*] Burak Cavdaroglu[*†] John E. Mitchell[‡]
Thomas C. Sharkey[*§] William A. Wallace[*†]

Abstract

We consider the problem of restoring services provided by infrastructure systems after an extreme event disrupts them. This research proposes a novel integrated network design and scheduling problem that models these restoration efforts. In this problem, work groups must be allocated to build nodes and arcs into a network in order to maximize the cumulative weighted flow in the network over a horizon. We develop a novel dispatching rule that selects the next *set* of tasks to be processed by the work groups. We further propose families of valid inequalities for an integer programming formulation of the problem, one of which specifically links the network design and scheduling decisions. Our methods are tested on realistic data sets representing the infrastructure systems of New Hanover County, North Carolina in the United States and lower Manhattan. These results indicate that our methods can be used in both real-time restoration activities and long-term scenario planning activities.

1 Introduction

The restoration of services provided by infrastructure systems is critical for society to recover from extreme events. Therefore, the managers of these systems are faced with demanding choices in formulating their restoration efforts after the event. This research proposes a novel class of integrated network design and scheduling problems that can be used to model the formulation of these restoration efforts. The operations of an infrastructure system can be modeled using a network-based representation where flows in the network model the services provided by the system and disruptions within it can be modeled as the removal of nodes and arcs from the network

[*]Department of Industrial and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180.

[†]The work of this author was supported by the US Department of Homeland Security under Award Number: 2008-ST-061-ND 0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the US Department of Homeland Security.

[‡]Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180.

[§]Corresponding author, e-mail: sharkt@rpi.edu.

(see, e.g., Lee et al. [10]). The restoration efforts associated with the system will focus on installing or repairing physical components within the system and can be modeled as installing nodes and arcs into the network. Therefore, we can view this selection of nodes and arcs as *network design* decisions. Traditional network design problems are often only concerned with the performance of the end design of the network; however, the driving performance metric, especially in the eyes of the public, in evaluating the restoration efforts is how well the services provided by the system come back online. This means that the network design decisions will be evaluated as they are being implemented, so that the scheduling decisions associated with them will have a significant impact on the objective. In particular, the performance of the network at time t, which is composed of the original network plus the nodes and arcs completed by t, will be evaluated by determining the (weighted) amount of flow that can be sent from supply nodes to demand nodes. This directly models the focus of the restoration efforts: it is on restoring services rather than on the monetary cost of the efforts.

The analysis of civil infrastructure systems is complex since they are interdependent (see O'Rourke [15]); disruptions in one can spread to others causing cascading failures (see Wallace et al. [22], Mendonca and Wallace [12], and Chang et al. [6]). Rinaldi et al. [17] note that managers of the infrastructure systems have become inclined to consider these interdependencies; however, the managers of a particular infrastructure will have little knowledge of the structure and operations of the other systems. We can expect that the managers of an individual infrastructure will understand the direct connections of it with other infrastructures and, therefore, can weigh the services provided to certain connections more heavily (e.g., a hospital has a higher weight than a residential household). Our class of problems can be used to model this situation and, further, explore the effects on the restoration plan of an infrastructure when we consider these interdependencies.

There has been some research done on problems related to our class of integrated network design and scheduling problems. Guha et al. [9] develop approximation algorithms for problems concerned with installing nodes into a power network to recover from disruptions; however, this work assumes that demand nodes simply need to be connected to supply nodes and thus does not model the capacity limitations of the system. Ang [3] develops an integer programming formulation of the problem of scheduling the installation of a set of nodes and arcs into a power network and applies it to problems of limited scale. This model requires that all nodes and arcs are installed meaning that it does not model the network design decisions often associated with the restoration efforts of an infrastructure after an extreme event. Xu et al. [24] apply a genetic algorithm to a problem associated with restoring power after an earthquake. The objective of this problem minimizes the average time that each customer is without power; therefore, this problem does not

prioritize demand to critical points within the infrastructure. Matisziw et al. [11] propose an integer programming model in order to restore networks where connectivity between pairs of nodes is the driving performance metric associated with the network. Cavdaroglu et al. [5] examined a model to determine the restoration efforts of a set of interdependent infrastructure systems. However, this work assumes that an individual infrastructure system fully understands the operations of the other systems and, therefore, may not be appropriate to apply to the restoration efforts of a single infrastructure.

This paper proposes an integrated network design and scheduling problem that can be applied to a variety of infrastructure systems. We propose an integer programming formulation of this problem and discuss valid inequalities for it that improve the effectiveness of solving it with a commercial software package. One of these inequalities specifically links the network design and scheduling decisions. This integer programming formulation can be useful in the long-term scenario planning activities for the managers of the infrastructure systems. These activities increase the preparedness of the managers and, therefore, will result in more effective decision-making after an actual extreme event. However, the time required to solve the integer programming formulation may prohibit it from being useful in the real-time restoration activities after an extreme event. Therefore, we propose a dispatching rule for this problem that integrates fundamental concepts from network flows (the residual network optimality conditions) and scheduling (the weighted shortest processing time dispatching rule). The rule focuses on selecting the next *set* of tasks to be processed by the resources rather than traditional dispatching rules that simply focus on selecting the next *individual* task to be processed. This dispatching rule is shown to be quite effective in practice and can be utilized in real-time restoration activities. Both the dispatching rule and integer programming formulation (with its valid inequalities) are tested extensively on realistic data sets representing infrastructure systems in New Hanover County, North Carolina in the United States and lower Manhattan in New York City. These results demonstrate the power of our proposed methods along with providing insight into the costs to the individual infrastructures in formulating their restoration efforts according to the priorities of the emergency managers of the regions.

The remainder of this paper is organized as follows. Section 2 provides the mathematical model of our integrated network design and scheduling problem and presents its initial integer programming formulation. Section 3 focuses on optimization methods to solve the problem including a dispatching rule (Section 3.1) and valid inequalities (Section 3.2). Section 4 applies our optimization methods for the integrated network design and scheduling problem on realistic case studies associated with infrastructure systems of New Hanover County, North Carolina. These case studies were created through collaborations with managers of these infrastructure systems and the

emergency manager of the county. Section 5 focuses on case studies associated with the power infrastructure of lower Manhattan. We provide concluding remarks in Section 6.

2 The Integrated Network Design and Scheduling Problem

The mathematical model of our integrated network design and scheduling (INDS) problem involves a network $G = (N, A)$ where N is the set of nodes and A is the set of arcs. The node set N and arc set A can be viewed as the 'operational network' immediately after the extreme event, i.e., these sets represent the components of the infrastructure that are unaffected by the event. There is a set of supply nodes, $S \subseteq N$, and a set of demand nodes, $D \subseteq N$. Each arc $(i, j) \in A$ has an associated capacity u_{ij} while each supply node $i \in S$ has a supply capacity s_i and each demand node $i \in D$ has a demand d_i. We are interested in sending flow (respecting the flow capacities of the arcs and the supply/demand capacities of the nodes) from the supply nodes to the demand nodes where each unit of flow that arrives at demand node $i \in D$ is given a weight of w_i. The performance of the network is evaluated by determining the maximum amount of weighted flow that can be sent from the supply nodes to the demand nodes. There is a set of arcs, A', that we can install into the network. Without loss of generality, this can model problems where we can install both nodes and arcs into the network since a node can be 'split' into two nodes and an arc (or two arcs). We are interested in scheduling a subset of the arcs in A' onto a series of parallel identical work groups, $k = 1, \ldots, K$, in order to install them into the network. The identical work group assumption is practical in the context of single infrastructure restoration since the differences between the work crews are often negligible with respect to the units used to measure the processing times. Each arc $(i, j) \in A'$ has an associated processing time, p_{ij}, and capacity, u_{ij}. We assume, without loss of generality, that the processing times are integral. We further assume that we are in a non-preemptive environment so that a task must be processed without interruption. We will evaluate the network at time t by determining the maximum weighted flow, which we denote by f_t, that can be sent through operational network, $G(t) = (N, A \cup A'(t))$ where $A'(t)$ is the set of arcs completed by time t. The objective function of our integrated network design and scheduling problem will then measure how well the network comes online, i.e., we will maximize

$$\sum_{t=1}^{T} \mu_t f_t,$$

where μ_t provides the weight we associate with the performance of the network at time t. The INDS problem was proven to be NP-hard even for problems with a single work group, single supply node, and single demand node in Nurre and Sharkey [14].

2.1 An Integer Programming Formulation

We now propose an integer programming formulation for the INDS problem. One of the difficulties with utilizing integer programming in order to solve the INDS problem is that modeling the sequencing decisions associated with the scheduling components of this problem requires a large number of variables. Typically, the sequencing decisions are either modeled with binary decisions variables representing the decision that task ℓ comes directly before task ℓ' on work group k or modeled with binary decision variables representing the decision that task ℓ is being completed by work group k at time period t. In our collaborations with the managers of the infrastructure systems in the areas represented by our data sets, we learned that typically the number of time periods in the problem is smaller than the number of potential arcs in A' (our set of tasks). Therefore, we have chosen to model the sequencing decisions with 'time-indexed' decision variables.

We further note that time-indexed formulations of scheduling problems lead to linear programming relaxations that are typically stronger than other formulations (e.g., Sousa and Wolsey [21] and Savelsbergh et al. [18]). Integer programming formulations and solution approaches of scheduling problems are discussed in Sousa and Wolsey [21], Schulz [19], Akker et al. [2], Waterer et al. [23], Mohring et al. [13], and Correa and Schulz [7]. The objective functions considered in these works are either makespan or weighted completion time, so it is only necessary to use variables that represent the completion times of the tasks. We will leverage these traditional time-indexed formulations by introducing variables to track if a task has already been completed and is, therefore, available in the network.

The variables in the integer programming formulation of the INDS problem can be broken down into three types of variables: (i) network flow variables, (ii) network design variables, and (iii) scheduling variables. The network flow variables include continuous variables x_{ijt} for $(i,j) \in A \cup A'$ and $t = 1, \ldots, T$ that represent the flow on arc (i,j) in time period t and continuous variables v_{it} for $i \in D$ that represent the amount of demand met at node i in time period t. The network design variables include binary variables β_{ijt} for $(i,j) \in A'$ and $t = 1, \ldots, T$ that represent that arc (i,j) is available in time period t. The scheduling variables include binary variables α_{kijt} for $k = 1, \ldots, K$, $(i,j) \in A'$, and $t = 1, \ldots, T$ that represent the decision that work group k completes arc (i,j) in time period t. The formulation of the INDS problem is:

$$\max \sum_{t=1}^{T} \sum_{i \in D} \mu_t w_i v_{it}$$

subject to (IP)

$$\sum_{(i,j) \in A \cup A'} x_{ijt} - \sum_{(j,i) \in A \cup A'} x_{jit} \leq s_i \qquad \text{for } i \in S,\ t = 1, \ldots, T \tag{1}$$

$$\sum_{(i,j)\in A\cup A'} x_{ijt} - \sum_{(j,i)\in A\cup A'} x_{jit} = 0 \quad \text{for } i \in N\backslash\{S\cup D\},\ t=1,\ldots,T \tag{2}$$

$$\sum_{(i,j)\in A\cup A'} x_{ijt} - \sum_{(j,i)\in A\cup A'} x_{jit} = -v_{it} \quad \text{for } i \in D,\ t=1,\ldots,T \tag{3}$$

$$0 \le v_{it} \le d_i \quad \text{for } i \in D,\ t=1,\ldots,T \tag{4}$$

$$0 \le x_{ijt} \le u_{ij} \quad \text{for } (i,j) \in A,\ t=1,\ldots,T \tag{5}$$

$$0 \le x_{ijt} \le u_{ij}\beta_{ijt} \quad \text{for } (i,j) \in A', t=1,\ldots,T \tag{6}$$

$$\sum_{(i,j)\in A'} \sum_{s=t}^{\min\{T,t+p_{ij}-1\}} \alpha_{kijs} \le 1 \quad \text{for } k=1,\ldots,K,\ t=1,\ldots,T \tag{7}$$

$$\beta_{ijt} - \beta_{ij(t-1)} = \sum_{k=1}^{K} \alpha_{kijt} \quad \text{for } (i,j) \in A',\ t=1,\ldots,T \tag{8}$$

$$\beta_{ij(t+1)} \ge \beta_{ijt} \quad \text{for } (i,j) \in A',\ t=1,\ldots,T-1 \tag{9}$$

$$\sum_{t=1}^{p_{ij}-1} \beta_{ijt} = 0 \quad \text{for } (i,j) \in A' \tag{10}$$

$$\sum_{k=1}^{K} \sum_{t=1}^{p_{ij}-1} \alpha_{kijt} = 0 \quad \text{for } (i,j) \in A' \tag{11}$$

$$\alpha_{kijt}, \beta_{ijt} \in \{0,1\} \quad \text{for } (i,j) \in A',\ k=1,\ldots,K,\ t=1,\ldots,T. \tag{12}$$

The objective is to maximize the cumulative weighted flow arriving at the demand nodes over the horizon of the problem. Constraints (1)-(6) are typical network flow constraints over the arcs available in the network in period t. They ensure that the flow generated at a supply node does not exceed its supply capacity (1), the amount of flow delivered to a demand node is equal to the satisfied demand at the node (3) while not exceeding the requested demand at the node (4), and the flow on an available arc does not exceed its capacity (5)-(6). Constraints (7)-(12) link the network design decisions with the scheduling decisions. We note that we have assumed that the performance of the network is evaluated at the end of the time period, so if $\alpha_{kijt} = 1$, then arc (i,j) is available in the network in time period t. Constraints (7) ensure that, at most, one task is being processed on work group k in time period t. This is because, if $\alpha_{kijs} = 1$, then the task corresponding to arc (i,j) will be processed during time periods $s - p_{ij} + 1$ through s on work group k. Constraints (8) ensure that if the arc (i,j) first becomes available in time period t (i.e., $\beta_{ijt} - \beta_{ij(t-1)} = 1$), then it must have been completed by some work group in that period. Constraints (9)-(11) are logical constraints that ensure an arc stays available once it is completed and that we do not complete an arc in a time period earlier than its required processing time.

The INDS problem is most applicable to single-commodity infrastructure systems which include, for example, power, water, waste water, and supply chain systems. The power system is

typically the backbone of civil infrastructure systems and is vulnerable to disruptions from many extreme events. However, the INDS problem assumes that we can directly control the flow in the network which is not the case for power systems (see Bienstock and Mattia [4]). The 'DC model' is a commonly used linear approximation of the power infrastructure (see the Appendix for its details) to model its operations. We will examine the INDS problem with the DC model in our computational testing. The INDS problem can be directly applied to water, waste water, and supply chain systems since the flow in these systems can be directly controlled. We note that the types of components in these systems that are vulnerable to damage will vary based upon the type of extreme event (e.g., the pipes of the waste water system are vulnerable to earthquakes while its pump stations are more vulnerable to flooding).

3 Solution Methods: Dispatching Rules and Valid Inequalities

This section is focused on the development and analysis of solution methods for the integrated network design and scheduling problem. We first develop a novel dispatching rule for the problem that selects the next *set* of arcs to be processed by the work groups. This dispatching rule will integrate fundamental concepts from the fields of network flows and scheduling. We then discuss families of valid inequalities for the integer programming formulation of the INDS problem.

3.1 Dispatching Rules

There has been a significant amount of research in parallel machine scheduling on so-called dispatching rules (Pinedo [16]). These rules often characterize the desirability of scheduling a certain task by estimating its contribution to the objective function and then greedily schedule the unscheduled task with the 'best' desirability. The key in developing a dispatching rule for our INDS problem is to understand how completing a task or, equivalently, an arc impacts the objective function of the problem. The installation of an arc (i, j) can improve the performance of the network by increasing the amount of weighted flow in it. This will then impact the objective function for the remainder of the horizon of the problem. We could attempt to develop a more traditional dispatching rule for our INDS problem by examining, for each uninstalled arc $(i, j) \in A'$, the improvement in the performance of the network by installing arc (i, j) into it. A modification of the classic weighted shortest processing time (WSPT) first rule (see Smith [20]) would then select and schedule the arc that maximizes the ratio of the improvement by installing the arc and the processing time of the arc. However, this is short-sighted in the sense that certain arcs will not increase the weighted flow if they are not performed in sequence with other arcs. For example, there may exist

a path of arcs from a supply node to a demand node that needs to be installed in order to increase the weighted flow. The installation of each of these arcs may not increase the weighted flow but the installation of this path may significantly increase the weighted flow. Therefore, our dispatching rule should be concerned with the contributions to the weighted flow of installing a *set* of arcs.

It is easiest to motivate the dispatching rule for the INDS problem by focusing on a special class of the problem where each demand node is given the same weight. In other words, the performance of the network is simply concerned with maximizing the total flow from the supply nodes to the demand nodes. We can assume that the performance of the network is evaluated by determining the maximum flow from a single supply node s to a single demand node τ. This assumption is without loss of generality by applying a standard network expansion technique to multiple supply/demand node networks. It is well-known that the residual network associated with a maximum flow in the network does not contain an augmenting path from the supply node s to the demand node τ (see Ahuja et al. [1]). Therefore, in order to increase the amount of flow sent from s to τ in the current operational network, we must install a set of arcs that form some residual path between the source and the sink node. Our dispatching rule for this INDS problem will then select a set of uninstalled arcs that belong to some residual path and that maximizes the ratio of the residual capacity of the path and the cumulative processing times of the uninstalled arcs in it. Mathematically, suppose that x^* is the current optimal flow in the network composed of the original arcs and the installed arcs from A'. The arcs in the residual network associated with x^* will have a residual capacity of r_{ij} and a processing time of $p_{ij} = 0$ since they are already installed. The uninstalled arcs in A' will have a residual capacity of u_{ij} (their original capacity) and a processing time of p_{ij} (their original processing time). We then define the residual capacity of path P as $r(P) = \min_{(i,j)\in P} r_{ij}$ and the processing time of a path as $p(P) = \sum_{(i,j)\in P} p_{ij}$. We are then interested in scheduling the uninstalled arcs in the path that is an optimal solution to the problem

$$\max_{P\in\Phi: p(P)>0} \frac{r(P)}{p(P)} \tag{13}$$

where Φ is the set of all paths from s to τ in the network composed of the residual network and all uninstalled arcs. We further note that that if $p(P) = 0$, this implies that all arcs are already installed in the network and that $r(P) = 0$ since x^* is the maximum flow in the network. The numerator of (13) provides a measure of the amount of additional flow in the network by installing arcs in P while the denominator provides a measure of the resources required to process P. For a single work group, $p(P)$ is precisely the 'makespan' required to complete all uninstalled arcs in the path. For multiple work groups, we could alter the denominator to approximate the makespan of the path by including the leading term $\frac{1}{K}$. This provides a lower bound on the actual makespan for the path. In terms of the optimization problem, this leading term is a constant for all paths, so

optimizing (13) is equivalent. Although this is only a lower bound on the makespan of a path for multiple work groups, the dispatching rule is shown to provide solutions of high-quality.

It is more difficult to determine an optimal path to (13) than it is to determine the next job according to the WSPT rule since we cannot decompose (13) by arcs. We will now discuss a combinatorial algorithm to determine an optimal solution to (13). The idea for the algorithm is motivated by the following observation: if we know that $r(P^*)$ is the numerator in an optimal solution to (13), then P^* is the path with the shortest processing time in the network where we only include arcs whose residual capacities are greater than or equal to $r(P^*)$. This immediately leads to an algorithm to solve (13): for each potential value of the numerator (i.e., the residual capacity of a path), we determine the shortest processing time path in the network containing only arcs whose residual capacities are above the numerator. An optimal solution is then the path obtained in this procedure that has the maximum ratio of residual capacity to processing time. We note that this procedure is easily adapted to situations where a constraint is placed on the denominator (for example, if we are in a single work group setting and at time t, we do not want to select a path with a processing time greater than $T - t$) by terminating the procedure when the shortest path in the network exceeds the threshold of the constraint. Note that the residual capacity of a path is the minimum residual capacity of the arcs in the path, so there are at most $2(|A| + |A'|)$ distinct values to be considered. This means that we can determine the next set of arcs to be processed by solving $O(|A| + |A'|)$ shortest path problems.

The dispatching rule for this INDS problem will determine the first set of tasks to be processed by solving the problem (13). We will assign these tasks (according to the longest processing time first rule) to the available work groups until all tasks from this set are processed. In other words, we can view the tasks that need to be processed as a queue and we will process the next task in the queue whenever a work group becomes available. If no tasks are in the queue, we will then determine the next set of arcs to be processed by considering the residual network associated with an optimal solution to the maximum flow problem where all arcs that are currently being processed are assumed to be available in the network. This process will continue until either all tasks are processed or we reach the end of the horizon.

We now discuss the dispatching rule for the INDS problem where the demand nodes can have different weights. In particular, we will select a residual path from some source node $j \in S$ to some demand node $i \in D$ that maximizes the ratio of w_i times the residual capacity of the path and the cumulative processing times of the uninstalled arcs in it. We can determine this set of uninstalled arcs by solving $O(|D|)$ problems of the form (13) - one for each distinct node $i \in D$, viewing that node as the 'super-demand' node τ. However, in the method for solving (13), note that for a fixed

residual capacity, we are solving a shortest path problem on the *same* set of arcs regardless of the node $i \in D$. Therefore, because of the structure of Dijkstra's algorithm, we can determine the relevant information (i.e., shortest path from the super-source node to each $i \in D$) for each of the $O(|D|)$ problems of the form (13) by solving $O(|A| + |A'|)$ shortest path problems. We will now formally present the algorithm to determine the next set of arcs to be processed according to the dispatching rule for the core INDS problem. This algorithm assumes that we are working with the network where we have created a 'super-supply' node s and have calculated the residual network, $G(x^*)$, associated with the current optimal weighted flow, x^*, in this network. We denote r_{ij} as the residual capacity of arc (i, j) where if arc (i, j) is uninstalled in the network $r_{ij} = u_{ij}$. The notation $G(N, A(r))$ is used to denote the network composed of only arcs with a residual capacity such that $r_{ij} \geq r$ where $(i, j) \in A \cup A'$ or $(j, i) \in A \cup A'$. Algorithm 1 provides the pseudo-code for determining the path selected by the dispatching rule.

Algorithm 1 Algorithm for Path Selection in the Dispatching Rule

Set MaxRatio $= 0$ and $P^* =$ null.
Sort the residual capacities of all arcs $(i, j) \in A \cup A'$ or $(j, i) : (i, j) \in A \cup A'$ in non-decreasing order and put them into array R.
for $\ell = 1, \ldots, 2(|A| + |A'|)$ **do**
    Determine the shortest path distance labels, $d(i, R[\ell])$ for $i \in D$, from the source node s in the network $G(N, A(R[\ell]))$.
    for all $i \in D$ **do**
        if $\frac{w_i R[\ell]}{d(i,R[\ell])} >$ MaxRatio **then**
            Set MaxRatio $= \frac{w_i R[\ell]}{d(i,R[\ell])}$.
            Set P^* to be the shortest path from s to i.
        end if
    end for
end for
Return P^*.

3.2 Valid Inequalities for IP Formulation

The purpose of this section is to present valid inequalities in order to strengthen the bounds provided by the linear programming relaxation of the IP formulation. We focus on four families of valid inequalities: (i) capacity inequalities, (ii) shortest processing time path inequalities, (iii) flow cover inequalities, and (iv) β-conservation inequalities. These families help to more closely link

the availability decisions with the flows in the network, which causes the β_{ijt} variables to increase to achieve the same level of flow on arc (i, j) in time period t. This, in turn, requires more resources to be dedicated to arc (i, j) prior to time period t. Therefore, these inequalities help to 'link' the flow variables with the scheduling decisions.

The capacity inequalities seek to tighten the capacity of an arc, u_{ij}, where $(i, j) \in A'$. It is clear that we could determine the maximum flow placed on arc (i, j) in any potentially feasible flow in the network $(N, A \cup A')$ and reduce u_{ij} to this value. However, if arc (i, j) belongs to a directed cycle, this value could be arbitrarily high since flow may move through this cycle without ever arriving at a demand node. We can thus restrict ourselves to feasible flows where every unit of flow reaches some demand node (i.e., the flow can be decomposed into a series of paths). Therefore, we can reduce u_{ij} to be equal to the minimum cut value that separates j from the set of demand nodes D, since every unit of flow on arc (i, j) must cross this cut to reach a demand node. We can further reduce the capacity of arc $(i, j) \in A'$ when all demand node weights are equal. We will assume that we are working with a network with a single supply and demand node. This further reduction focuses on the additional flow sent through the network by installing some subset of arcs in A'. It relies on the following result.

Theorem 3.1 *There exists a maximum flow in the network $\bar{G} = (N, A \cup \bar{A})$, where $\bar{A} \subseteq A'$ such that the flow on arc $(i, j) \in \bar{A}$ is less than or equal to $\bar{v} - v$ for all $(i, j) \in \bar{A}$, where $\bar{v}$ is the maximum flow in the network $\bar{G}$ and v is the maximum flow in the network G.*

Proof: We will construct a solution that satisfies this property be applying the augmenting path algorithm (see Ahuja et al. [1]) starting from the maximum flow in G in order to determine the maximum flow in $\bar{G}$. The proof will be by induction on iteration ℓ of the augmenting path algorithm. We let x_{ij}^{ℓ} denote the flow on arc (i, j) and $\bar{v}^{\ell}$ represent the flow into the demand node after iteration ℓ of the algorithm. It is clear that for $\ell = 0$, that $x_{ij} \leq \bar{v}^{\ell} - v$ for all $(i, j) \in \bar{A}$. We assume the claim holds up to iteration ℓ and show it holds for $\ell + 1$. Let P be the augmenting path found in iteration $\ell + 1$. By definition, the algorithm pushes $\bar{v}^{\ell+1} - \bar{v}^{\ell}$ flow along this path. The only way for the flow on arc $(i, j) \in A'$ to increase is for it to be part of this path. For this type of arc, we know that $x_{ij}^{\ell+1} = x_{ij}^{\ell} + (\bar{v}^{\ell+1} - \bar{v}^{\ell}) \leq \bar{v}^{\ell} - v + \bar{v}^{\ell+1} - \bar{v}^{\ell} = \bar{v}^{\ell+1} - \bar{v}$. Our desired result holds by induction. □

Theorem 3.1 holds for any set $\bar{A} \subseteq A'$. We can, therefore, reduce the capacity of arc $(i, j) \in A'$ to $v' - v$ where v' is the maximum flow in $(N, A \cup A')$.

The shortest processing time path constraints are a generalization of constraints (10) where we also consider the time required to build a path of arcs to node i. In particular, we know that the

flow on arc $(i,j) \in A'$ cannot be positive until we install some path to node i and arc (i,j). We are, therefore, concerned with the shortest processing time path from a supply node to node i plus the process time of arc (i,j). We then know that arc $(i,j) \in A'$ cannot have flow on it until this value divided by the number of work groups (i.e., the lower bound on the makespan to complete this path and arc (i,j)). This helps rule out 'partial' installations of a path to arc (i,j) before this time in the linear programming relaxation.

The flow cover inequalities of Gu et al. [8] can be applied to the network during each time period t for each arc $(i,j) \in A'$ and each cutset $C(i)$ separating all supply nodes from i, provided that the cutset contains only arcs in A', i.e., $C(i) \subseteq A'$. This would be especially applicable, for example, in disruptions to infrastructures that wipe out entire portions of the network. These inequalities link the flow on arc (i,j) in time period t with the availability of arcs in the cutset, since any flow on arc (i,j) must be on some arc in $C(i)$. We can define these inequalities as

$$x_{ijt} \leq u_{ij} \sum_{(\kappa,\iota) \in C(i)} \beta_{\kappa\iota t} \text{ for } t = 1, \ldots, T. \tag{14}$$

Let $O(i)$ be the set of nodes such that there exists an arc $(\kappa, i) \in A'$. If the set of arcs $\{(\kappa, i) : \kappa \in O(i)\}$ is a valid choice for $C(i)$ then the corresponding inequality focuses on those arcs that are 'one step back' from node i in the network, i.e., they come directly into node i. We can apply the same type of inequalities to arcs that are 'two steps back' from node i in the network, i.e., those arcs that come into some node $\kappa \in O(i)$, as long as all arcs two steps back are in A'. This family of inequalities can be extended to an arbitrary number of steps back from node i as long as all arcs that far back are in A'.

The β-conservation inequalities are motivated by the fact that if arc (i,j) is operational (e.g., has flow on it), then some arc into node i must also be operational. They are similar to the flow-cover inequalities since they can only be applied to an arc $(i,j) \in A'$ where the only arcs coming into node i belong to the set A'. These inequalities are

$$\beta_{ijt} \leq \sum_{\kappa \in O(i)} \beta_{\kappa i t} \text{ for } t = 1, \ldots, T. \tag{15}$$

These constraints help link the scheduling decisions of a path of arcs by forcing those that are earlier in the path to be installed prior to the arcs later in the path. However, these constraints together with constraints (8) may lead to unnecessary restrictions for multiple work groups if the processing time of arcs later in the path are longer than those earlier in the path. Therefore, we will modify the 'definition' of β_{ijt} to be that arc $(i,j) \in A'$ can be operational (i.e., it is completed but

may or may not have flow on it), so constraints (8) become

$$\beta_{ijt} \leq \sum_{k=1}^{K} \sum_{s=1}^{t} \alpha_{kijs} \text{ for } (i,j) \in A', \, t = 1, \ldots, T. \tag{16}$$

This small change allows us to add in the β-conservation inequalities into the IP formulation. Note that we could also apply the flow cover inequalities to this modified IP formulation. We now prove a relationship between these inequalities.

Theorem 3.2 *The flow cover inequalities for one-step and two-step back cutsets are implied by the β-conservation inequalities, if the network has no parallel arcs.*

Proof: Note that

$$x_{ijt} \leq u_{ij}\beta_{ijt} \leq u_{ij} \sum_{\kappa \in O(i)} \beta_{\kappa it},$$

which shows that the β-conservation inequalities imply the one step back flow cover inequalities. For the two step back flow cover inequalities, we apply the β-conservation inequalities to each node $\kappa \in O(i)$ to yield

$$u_{ij} \sum_{\kappa \in O(i)} \beta_{\kappa it} \leq u_{ij} \sum_{\kappa \in O(i)} \sum_{q \in O(\kappa)} \beta_{q\kappa t},$$

where no β term is repeated because there are no parallel arcs. Note that if there was some arc into $\kappa \in O(i)$ that is in A, the two step back inequalities would not be valid for arc (i,j) as well as the β-conservation constraint for κ. □

4 Computational Testing on the New Hanover County Data Set

We will now discuss the results of case studies of applying the INDS problem to realistic data sets representing infrastructure systems in New Hanover County, North Carolina in the United States. New Hanover County is a coastal county in southern North Carolina that includes the city of Wilmington and the Cape Fear beaches. These data sets were created through extensive collaborations with the managers of the infrastructure systems in New Hanover County as well as collaborations with the emergency manager of the county. This section focuses on applying the INDS problem to three separate infrastructure systems: (i) the power infrastructure of the county, (ii) the waste water infrastructure of the county, and (iii) an emergency supply chain infrastructure in the city of Wilmington. The disruptions to the power and waste water infrastructures model the effects of a strong hurricane whose eye passes to the south of the county so that both extensive

flooding and wind damage are possible, which is similar to the scenario caused by Hurricane Ophelia in Septemeber 2005. The emergency supply chain infrastructure models situations in which the emergency manager of the county must deliver critical goods to the affected population in Wilmington that were unable to leave the city prior to the hurricane.

In the analysis of these case studies, we are interested in examining the performance of the dispatching rule and the integer programming formulation of the INDS problem. It is especially important to determine the potential for these methods to be used in real-time restoration activities. Therefore, our testing was performed on a laptop with a 2.16 GHz Intel Core 2 Duo Processor with 3 GB of RAM, which would be similar to the computing resources available during real-time restoration activities. We have used CPLEX 12.0 in order to solve the integer programming formulation of the INDS problem. We have chosen to only examine the initial integer programming formulation of the INDS problem since it is solved rather quickly and the disruptions do not wipe out large portions of the network (so the flow cover and β-conservation inequalities are not applicable). It is also important to explore the effects on the restoration plan of a certain infrastructure when we consider its interdependencies with other infrastructures. This is especially important for New Hanover County since the emergency manager of the county has certain priorities in the overall recovery efforts of the county. Therefore, we will explore the effects on the restoration plan of a particular infrastructure when it is formed according to the priorities of the county instead of the priorities of the infrastructure. The priorities of the emergency manager of the county, in decreasing order of their importance, are: (1) emergency communications centers, (2) hospitals and emergency shelters, (3) police and fire departments, (4) all components in the water and wastewater infrastructures, and (5) all other components (including residential households).

4.1 Power Infrastructure Case Study

The focus of this section is on examining a case study representing the power infrastructure of New Hanover County. The network model of this infrastructure has 377 nodes and 386 arcs under normal operations. It may seem that there is not much redundancy in the system but we note that there is a good level of redundancy in the *transmission* network in the county. The managers of the power infrastructure suggested that the model should focus on the transmission network of the county - so that the distribution networks to the demand points are simplistic. There are 37 nodes and 46 arcs in the transmission network. We have added 340 demand nodes and 340 arcs connecting these nodes to appropriate distribution substations in our model of this infrastructure. These demand nodes and arcs then model the distribution network in the county. The INDS problem has $|N| = 377$ nodes, $|A| = 346$ arcs, and $|A'| = 40$ arcs as network design decisions. The horizon

of the problem is equal to $T = 30$ where, roughly, each time period represents a six hour block of time so that the horizon is roughly a week. We consider two different types of weights on the performance of the network (i.e., μ_t for $t = 1, \ldots, T$) where the first class ('Constant') weighs the performance evenly over the horizon and the second class ('Scaled') weighs the performance more heavily later in the horizon by setting $\mu_t = t/T$. We further consider two different types of weight for the demand nodes: one where each unit of met demand is weighed evenly across the demand nodes ('Constant') and one where the demand nodes with higher priorities according to the emergency manager of the county have larger weights ('Priority'). The number of work groups in this study is equal to $K = 1$ or $K = 2$ which represents the fact that the county itself has very limited resources.

Table 1 provides the computational performance of the dispatching rule and the integer programming formulation of the INDS problem. Note that the gap of the various solution methods was 'normalized' by setting the performance of the network in each time period equal to its maximum weighted flow minus the maximum weighted flow in the network without any arcs from A'. Therefore, the initial performance of the network does not bias the gap of the solution methods. These results indicated that the dispatching rule obtains high-quality (and optimal in some instances) solutions to the INDS problem extremely quickly. CPLEX 12.0 is able to identify an optimal solution to the problem quickly as well, although commercial software packages may not be available to managers of the infrastructure systems during real-time restoration activities.

			Dispatching Rule		IP	
K	μ_t	w_i	Time (s)	Opt. Gap	Time (s)	Opt. Gap
1	Constant	Constant	0.86	0.00%	3.13	0.00%
1	Constant	Priority	0.93	0.00%	3.15	0.00%
1	Scaled	Constant	0.86	0.00%	3.24	0.00%
1	Scaled	Priority	0.93	0.00%	3.26	0.00%
2	Constant	Constant	0.91	0.12%	3.09	0.00%
2	Constant	Priority	0.97	0.12%	3.16	0.00%
2	Scaled	Constant	0.90	0.14%	3.15	0.00%
2	Scaled	Priority	0.98	0.12%	3.21	0.00%

Table 1: The performance of the dispatching rule and IP formulation on the power infrastructure case study.

The restoration efforts formulated by the dispatching rule and the INDS problem can assist in the decision-making of the managers of the power infrastructure. However, these managers

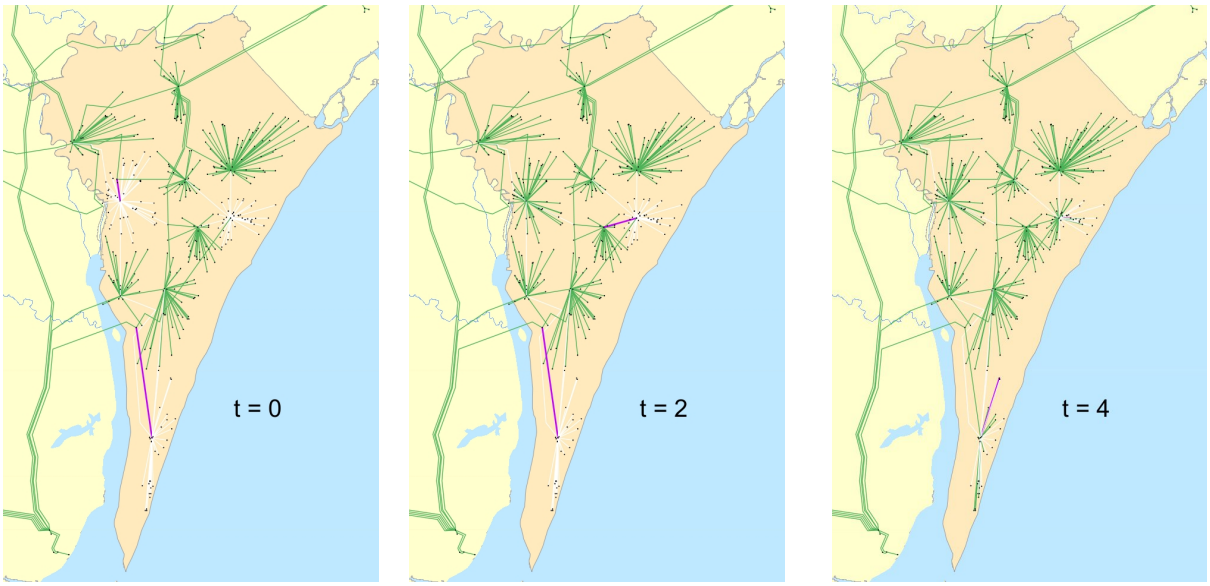

Figure 1: Scheduling/resource allocation decisions (purple arcs) and the performance of the network (green arcs have flow on them) for optimal restoration efforts in the power infrastructure of New Hanover County.

may have limited mathematical expertise implying that it would be beneficial to reproduce the restoration efforts using visualization tools. Figure 1 provides an example of such a visualization tool for the optimal restoration efforts for the problem with $K = 2$, Constant μ_t, and Priority demand node weights. This visualization tool was produced using a geographic information system (GIS) to display data on the infrastructure.

The 'Constant' weights for the demand nodes can be viewed as modeling the objective of the managers of the power infrastructure. This is because a unit of met demand generates similar revenues regardless of the demand node where it is delivered. Therefore, the operations of the infrastructure under the performance metric of the managers would be to deliver the maximum amount of power possible throughout the system. It is clear that the 'Priority' weights for the demand nodes reflect the performance metric that aligns the restoration efforts of the power infrastructure with the overall goals of the recovery efforts of the emergency manager of the county. We have observed that, for all combinations of work groups (K) and performance weights (μ_t), the optimal restoration plan aligned with the priorities of the emergency manager is also one of the optimal solutions to the problem whose objective function represents the focus of the managers of the power infrastructure. Therefore, the power infrastructure can align its restoration efforts with the overall goals of the recovery effort without any effect on its own objective.

There is a potential issue in applying the restoration efforts obtained either through the dis-

patching rule or integer programming formulations of the INDS problem: it was obtained with a primarily 'flow-based' model of the operations of the infrastructure. As discussed in the Appendix, this is not the case for the power infrastructure since power flows according to certain physical laws. Therefore, we will explore the quality of restoration efforts (i.e., which arcs to process and when to process them) obtained through these methods when the operations of the power infrastructure are modeled according to the DC model discussed in the Appendix. The performance of the network in each time period for the 'scheduling' solution returned by the dispatching rule is calculated through the DC model over the available arcs in the network. We, therefore, associate a value $v(\text{DR})$ with the dispatching rule, which represents the cumulative performance of the DC model applied to the available network over the horizon of the problem. The gap of the dispatching rule is then calculated as

$$100\% * \frac{v^*(\text{DC}) - v(\text{DR})}{v(\text{DR})},$$

where $v^*(\text{DC})$ is the optimal objective function value to the INDS problem with the DC model. The gap associated with the optimal scheduling solution to the core INDS problem was calculated in a similar fashion. Table 2 presents the performance of the restoration efforts returned by the dispatching rule and the optimal restoration efforts in the core INDS problem when compared with the optimal restoration efforts in the INDS problem with the DC network model. The performance of the restoration efforts obtained using the core INDS problem and the dispatching rule perform extremely well under the DC model of the problem. We further note that, for all combinations of work groups and performance weights the optimal solution to the INDS problem with the DC model and weighted demand nodes is also an optimal solution to the same problem with constant demand node weights.

4.2 Waste Water Infrastructure Case Study

The focus of this section is on a case study of the waste water infrastructure in New Hanover County. The initial network model of this infrastructure has 543 nodes and 538 arcs. There is no redundancy in this infrastructure - it is composed of 5 separate components, which would correspond to trees in the network representation. Each of these components has a 'root' node that is a treatment plant and all arcs in a particular component are directed away from the plant. This may seem counter-intuitive since the waste water actually flows toward the treatment plant. However, modeling the infrastructure in this fashion allows us to view the requested service at a particular node as demand in the network representation. The disruption scenario in this case study represents damage to 25 pump stations that could occur simultaneously with the disruption scenario for

			Dispatching Rule		INDS Problem		INDS Problem with DC Model	
K	μ_t	w_i	Time (s)	Opt. Gap	Time (s)	Opt. Gap	Time (s)	Opt. Gap
1	Constant	Constant	0.86	2.98%	3.13	2.98%	3.83	0.00%
1	Constant	Priority	0.93	3.14%	3.15	3.14%	3.90	0.00%
1	Scaled	Constant	0.86	1.07%	3.24	1.07%	3.91	0.00%
1	Scaled	Priority	0.93	1.21%	3.26	1.21%	4.00	0.00%
2	Constant	Constant	0.91	1.42%	3.09	0.14%	8.60	0.00%
2	Constant	Priority	0.97	1.53%	3.16	0.28%	8.72	0.00%
2	Scaled	Constant	0.90	0.36%	3.15	0.05%	8.01	0.00%
2	Scaled	Priority	0.98	0.39%	3.21	0.09%	7.11	0.00%

Table 2: The performance of the restoration efforts obtained through the dispatching rule and the core INDS problem under the DC model.

the power infrastructure from Section 4.1. The pump stations are nodes in the network representation of the infrastructure, however, since the infrastructure is composed of 5 trees and all arcs are directed away from the root node of the tree, each node has one incoming arc into it. Therefore, we can model the disruption to the pump station as damage to the incoming arc into it. Therefore, we have that $|N| = 543$, $|A| = 513$, and $|A'| = 25$ in this case study. We note that, since the damage occurs at a pump station of the infrastructure, it is necessary to fix that pump station rather than implement some alternative route. Therefore, the design decisions in A' correspond exactly to the arcs representing the damaged pump stations. The horizon of the problem is set equal to $T = 30$, where each time period represents a roughly six hour block of time.

Table 3 provides the computational performance of the dispatching rule and the integer programming formulation for the INDS problem. We again consider two types of performance weights (μ_t) and demand node weights (w_i). These computational results are very similar to the results for the power infrastructure case study: the dispatching rule provides high-quality solutions in seconds and can thus be used in real-time activities. Further, the application of CPLEX 12.0 to the integer programming formulation of the INDS problem provides an optimal solution to the problem in real-time. We again investigate the effects on the restoration plan of the waste water infrastructure by formulating it considering the priorities of the emergency managers. This investigation demonstrated that the optimal restoration plan to the problem with priority-based weights for the demand nodes is also an optimal restoration plan to the problem with constant weights for the demand nodes. Therefore, this indicates that, for this case study, the managers of the waste water infrastructure can align their restoration efforts with the goals of the overall recovery ef-

fort without any effect to their objective function. Figure 2 provides a visualization tool for the restoration efforts of the problem with $K = 2$ work groups, Constant μ_t, and Priority demand node weights for the waste water infrastructure.

			Dispatching Rule		IP	
K	μ_t	w_i	Time (s)	Optimality Gap	Time (s)	Optimality Gap
1	Constant	Constant	1.24	1.80%	2.65	0.00%
1	Constant	Priority	1.33	2.05%	2.61	0.00%
1	Scaled	Constant	1.24	1.33%	2.60	0.00%
1	Scaled	Priority	1.33	1.46%	2.64	0.00%
2	Constant	Constant	1.44	0.94%	2.47	0.00%
2	Constant	Priority	1.53	1.10%	2.44	0.00%
2	Scaled	Constant	1.45	0.42%	2.46	0.00%
2	Scaled	Priority	1.53	0.47%	2.46	0.00%

Table 3: The performance of the dispatching rule and IP formulation on the waste water infrastructure case study.

4.3 Emergency Supply Chain Infrastructure Case Study

The focus of this section is on a case study corresponding to setting up an emergency supply chain infrastructure in New Hanover County. This emergency supply chain will deliver a critical good (e.g. food or water) to those affected by the hurricane that did not have the resources to evacuate the area prior to its landfall. The supply chain will set up a number of distribution sites in lower income areas. Therefore, this case study focuses primarily on setting up distribution sites in the city of Wilmington since all other areas have a median income of over $25,000 dollars. The potential locations for these distribution sites correspond to well-known, central entities in the city (such as malls, schools, and parks) and were determined through input from the emergency manager of the county. In this case study, these critical goods will flow into the county through the Red Cross, be routed to the operational distribution sites, and then 'delivered' to the affected population that come to the sites. The network model for the INDS problem representing this case study will have two nodes (say i and i') and an arc ((i, i') in A') representing each potential distribution site. There will be an arc from the Red Cross to each potential distribution site and then arcs from each potential distribution site to the different populations to which that site can deliver goods. The demand for these different populations was created using census tract information and the arcs

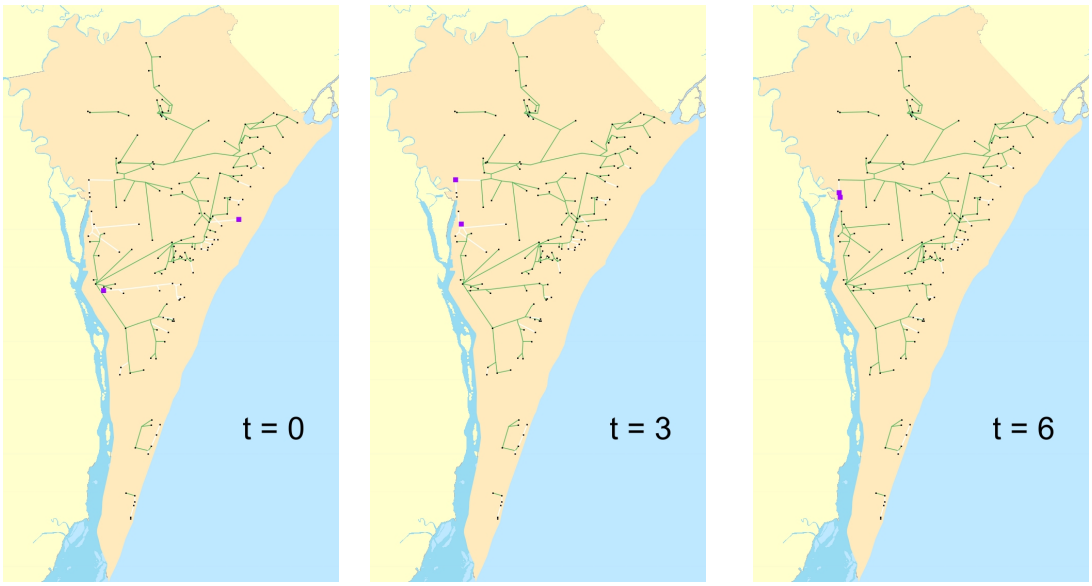

Figure 2: Scheduling/resource allocation decisions (purple nodes) and the performance of the network (green arcs have flow on them) for optimal restoration efforts in the waste water infrastructure of New Hanover County.

from the potential sites to the populations were included based on whether the site was close to the population. This case study has $|N| = 59$ nodes, $|A| = 85$ arcs, and $|A'| = 23$ design decisions. The problem has a horizon of $T = 12$, where each time period corresponds to 6 hours. Table 4 presents the computational performance of the various solution methods on this case study for problems where all demand is weighted equally. The dispatching rule does provide good quality solutions extremely quickly. CPLEX 12.0 was able to solve the integer programming formulation of this problem in real-time as well. Figure 3 provides a visualization tool for the restoration efforts of the problem with $K = 2$ work groups, Constant μ_t, and Constant demand node weights for the emergency supply chain infrastructure. This map focuses specifically on the city of Wilmington.

5 Computational Testing on the Lower Manhattan Data Set

We will now discuss the results of a case study of applying the INDS problem to a realistic data set representing the power infrastructure of lower Manhattan in New York City. We will consider a case study which represents the effects of the disruption scenario of the failure of components in and around the Brooklyn-Battery tunnel. Therefore, this disruption scenario has a large portion of the network wiped out, meaning the valid inequalities presented in Section 3.2 are applicable. This data set was created through close collaborations with officials in this system and was first

			Dispatching Rule		IP	
K	μ_t	w_i	Time (s)	Optimality Gap	Time (s)	Optimality Gap
1	Constant	Constant	0.007	2.92%	2.88	0.00%
1	Scaled	Constant	0.007	2.63%	1.07	0.00%
2	Constant	Constant	0.021	4.71%	9.82	0.00%
2	Scaled	Constant	0.020	7.85%	4.64	0.00%

Table 4: The performance of the dispatching rule and IP formulation on the emergency supply chain infrastructure case study.

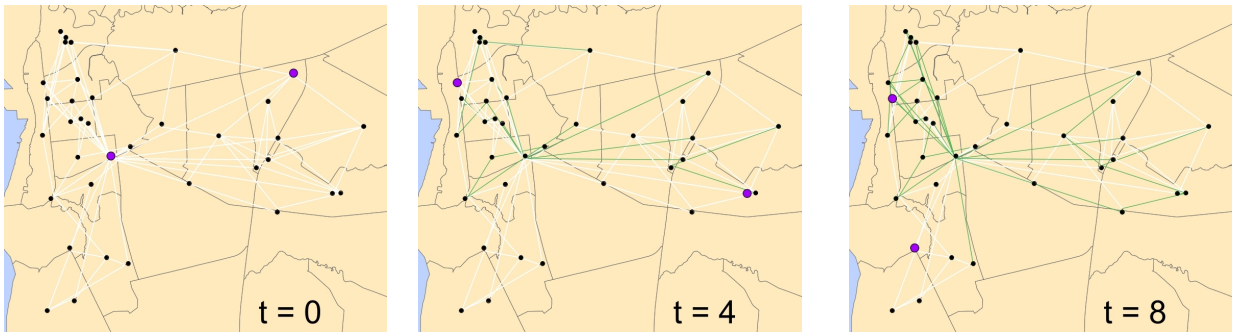

Figure 3: Scheduling/resource allocation decisions (purple nodes) and the performance of the network (green arcs have flow on them) for optimal restoration efforts in the emergency supply chain infrastructure in Wilmington.

presented in the work of Lee et al. [10]. The network in the INDS problem has $|N| = 1810$ nodes and $|A| = 2621$ arcs and is significantly larger than the networks from the case studies considered in Section 4. Further, the number of design alternatives is $|A'| = 695$ arcs. We have considered problems with $K = 1, 2, 3$ work groups, where the latter size is representative of the resources considered in Lee et al. [10]. The horizon of the INDS problem is $T = 60$ time periods.

Table 5 presents the computational results for the dispatching rule and initial integer programming formulation for the lower Manhattan data set. The results indicate that CPLEX 12.0 is not capable of determining (or verifying) the optimal solution to the problem within a six hour time limit for the initial integer programming formulation of the problem. We note that we have chosen to provide CPLEX 12.0 with the restoration plan determined by the dispatching rule as a warm-start solution. The gap of the dispatching rule is an *upper bound* on its actual gap, since it was calculated by comparing the objective function of the solution with the best known upper bound identified for the instance across all computational testing in this section. It is important to note that CPLEX 12.0 did not identify a better solution than the one provided by the dispatching rule for any instance of the problem in the six hour time limit. This means that the dispatching rule is more appealing to apply to the INDS problem than CPLEX 12.0 for these large-scale problems. Note that, due to the high running times of the core INDS problem, we have not examined the INDS problem with the DC Model for this case study since it is imperative in that analysis to obtain the optimal solution to the INDS problem with the DC model.

			Dispatching Rule		IP	
K	μ_t	w_i	Time (s)	Opt. Gap Upper Bound	Time (s)	Opt. Gap
1	Constant	Constant	5.54	16.52%	21600	24.48%
1	Scaled	Constant	5.55	10.73%	21600	13.64%
2	Constant	Constant	4.80	6.87%	21600	8.47%
2	Scaled	Constant	4.80	2.42%	21600	2.90%
3	Constant	Constant	4.18	4.51%	21600	5.53%
3	Scaled	Constant	4.18	1.22%	21600	1.40%

Table 5: The performance of the dispatching rule and IP formulation on the power infrastructure case study of lower Manhattan.

We now explore the effectiveness of the valid inequalities presented in Section 3.2 on this case study. The capacity inequalities do not effect the number of constraints or variables in the formulation, so we view them as 'standard' for the other inequalities. For this case study, we implemented the capacity inequalities based on lowest value obtained through each of the three techniques pre-

sented in Section 3.2. Table 6 presents the solution time and percentage improvement in the linear programming relaxation for each of the remaining inequalities plus the capacity inequalities. The shortest processing time path inequalities do not add much in terms of computation time, so we have also explored them in combination with the flow cover inequalities and β-conservation inequalities in Table 7. These results demonstrate the tradeoff between the quality of inequalities and the solution time required to solve the relaxation. Therefore, we tested both these classes of valid inequalities on the integer programming formulation for the full six hour time limit. These results are presented in Table 8.

			IP Relaxation	Capacity		Shortest Path + Capacity		Two Step Flow Cover + Capacity		β-conservation + Capacity	
K	μ_t	w_i	Time (s)	Time (s)	Improvement	Time (s)	Improvement	Time (s)	Improvement	Time (s)	Improvement
1	Constant	Constant	58.23	410.48	1.53%	281.03	6.93%	1604.07	9.49%	16984.54	10.72%
1	Scaled	Constant	46.61	235.94	0.29%	209.36	1.56%	1799.67	2.92%	17154.55	3.40%
2	Constant	Constant	30.84	75.91	0.00%	82.19	0.08%	1142.12	2.02%	8184.34	2.45%
2	Scaled	Constant	30.88	90.30	0.00%	78.95	0.02%	959.58	0.43%	13157.84	0.53%
3	Constant	Constant	34.02	69.20	0.00%	85.72	0.00%	972.20	0.18%	5370.55	0.32%
3	Scaled	Constant	34.15	60.19	0.00%	86.70	0.00%	1146.35	0.03%	5827.52	0.06%

Table 6: Strength of the valid inequalities on the linear programming relaxation.

			IP Relaxation	Two Step Flow Cover + Capacity + Shortest Path		β-conservation + Capacity + Shortest Path	
K	μ_t	w_i	Time (s)	Time (s)	Improvement	Time (s)	Improvement
1	Constant	Constant	58.23	1326.27	11.06%	9420.33	12.17%
1	Scaled	Constant	46.61	1309.95	3.22%	6051.16	3.70%
2	Constant	Constant	30.84	248.83	2.03%	11527.58	2.52%
2	Scaled	Constant	30.88	254.28	0.43%	8761.33	0.54%
3	Constant	Constant	34.02	1015.03	0.18%	4743.86	0.32%
3	Scaled	Constant	34.15	1019.92	0.03%	5407.26	0.06%

Table 7: Strength of the two step flow cover and β-conservation inequalities in combination with the capacity and shortest processing time path inequalities on the linear programming relaxation

We note that we have applied CPLEX 12.0 to the IP formulation with the capacity, shortest path, and β-conservation inequalities for the problem with $K = 3$ and constant μ_t on an eight-core computer. After a week of computational time, CPLEX 12.0 still had an optimality gap of 4.12% and had not identified a better solution than the one obtained by the dispatching rule. It is unreasonable to expect the managers of the infrastructure systems to have access to this type of

			IP		Two Step Flow Cover + Capacity + Shortest Path		β-conservation + Capacity + Shortest Path	
K	μ_t	w_i	Time (s)	Opt. Gap	Time (s)	Opt. Gap	Time (s)	Opt. Gap
1	Constant	Constant	21600	24.48%	21600	16.52%	21600	18.19%
1	Scaled	Constant	21600	13.64%	21600	10.73%	21600	11.42%
2	Constant	Constant	21600	8.47%	21600	6.87%	21600	7.42%
2	Scaled	Constant	21600	2.90%	21600	2.42%	21600	2.79%
3	Constant	Constant	21600	5.53%	21600	4.51%	21600	4.61%
3	Scaled	Constant	21600	1.40%	21600	1.23%	21600	1.22%

Table 8: Computational results for the two step flow cover and β-conservation in combination with the capacity and shortest processing time path inequalities on the INDS problem.

computing resource and be willing to wait for over a week to determine their restoration plan, even in their scenario planning activities. Therefore, the dispatching rule is an important tool for the restoration of larger infrastructure systems.

6 Conclusions

This research has developed a novel integrated network design and scheduling problem that can be used to model the problem of restoring services provided by infrastructure systems after an extreme event disrupts them. The model is general enough to be applicable to a variety of infrastructures, including the power, water, waste water, and emergency supply chain infrastructures. We have developed a novel dispatching rule for the INDS problem that focuses on selecting a set of arcs to process by examining the residual path optimality conditions from the area of network optimization. This dispatching rule has been shown to provide near-optimal solutions to realistic case studies in seconds, for many different infrastructure systems, and can thus be used in real-time restoration activities. We have further developed an integer programming formulation of the INDS problem that was able to provide the optimal solutions to the smaller case studies quickly. However, the application of CPLEX 12.0 to this integer programming formulation for the larger case studies is not able to identify an optimal solution to the problem even when using significant computational resources.

This research has applied the optimization models and algorithms developed for the INDS problem to several realistic case studies representing infrastructure systems in New Hanover County in North Carolina and lower Manhattan in New York City. The network models of the infrastruc-

ture systems in these case studies were created through careful collaborations with the managers of these systems. The disruptions/damage scenarios in the New Hanover County case studies were created to represent the effects of a hurricane whose eye passed just to the south of the county, implying that the strongest winds and flooding associated with the hurricane will affect the county. For the power infrastructures and waste water infrastructure, we examined the effects of the restoration efforts of the infrastructures when they are aligned with the priorities of the emergency manager of the county. The case studies indicate that the restoration efforts of these infrastructures can be aligned with these priorities with no detrimental effects to the objective of the infrastructures. These case studies demonstrate that the dispatching rule can serve as a powerful decision support tool in real-time restoration activities. It will be important, in the future, to integrate the dispatching rule and integer programming methods for the INDS problem into decision support systems for managers of infrastructure systems.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: Theory, algorithms, and applications*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[2] J.M. van der Akker, C.P.M. van Hoesel, and M.W.P. Savelsbergh. A polyhedral approach to single-machine scheduling problems. *Mathematical Programming*, 85(3):541–572, 1999.

[3] C.C. Ang. Optimized recovery of damaged electrical power grids. Master's thesis, Naval Postgraduate School, 2006.

[4] D. Bienstock and S. Mattia. Using mixed-integer programming to solve power grid blackout problems. *Discrete Optimization*, 4(1):115–141, 2007.

[5] B. Cavdaroglu, E. Hammel, J.E. Mitchell, T.C. Sharkey, and W.A. Wallace. Integrating restoration and scheduling decisions for disrupted interdependent infrastructure systems. Technical report, Department of Industrial and Systems Engineering, Rensselaer Polytechnic Insitute, 2009.

[6] S.E. Chang, T.L. McDaniels, J. Mikawoz, and K. Peterson. Infrastructure failure interdependencies in extreme events: Power outage consequences in the 1998 ice storm. *Natural Hazards*, 41(2):337–358, 2007.

[7] J.S. Correa and A. Schulz. Single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30(4):1005–1021, 2005.

[8] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming*, 85(3):439–467, 1999.

[9] S. Guha, A. Moss, J.S. Naor, and B. Schieber. Efficient recovery from power outage. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 1999.

[10] E.E. Lee, J.E. Mitchell, and W.A. Wallace. Restoration of services in interdependent infrastructure systems: A network flows approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1303–1317, 2007.

[11] T.C. Matisziw, A.T Murray, and T.H. Grubesic. Strategic network restoration. *Networks and Spatial Economics*, 10(3):345–361, 2010.

[12] D. Mendonca and W.A. Wallace. Impacts of the 2001 World Trade Center attack on New York City critical infrastructures. *Journal of Infrastructure Systems*, 12(4):260–270, 2006.

[13] R. Möhring, A.S. Schulz, F. Stork, and M. Uetz. Solving project scheduling problems by minimum cut computations. *Management Science*, 49(3):330–350, 2003.

[14] S.G. Nurre and T.C. Sharkey. Restoring infrastructure systems: An integrated network design and scheduling problem. In *Proceedings of the 2010 Industrial Engineering Research Conference*.

[15] T.D. O'Rourke. Critical infrastructure, interdependencies, and resilience. *The Bridge: National Academy of Engineering*, 37(1):22–29, 2007.

[16] M.L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, New York, 2008.

[17] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems Magazine*, 21(6):11–25, 2001.

[18] M.W.P. Savelsbergh, R.N. Uma, and J. Wein. An experimental study of LP-based approximation algorithms for scheduling problems. *INFORMS Journal on Computing*, 17(1):123–136, 2005.

[19] A.S. Schulz. *Polytopes and scheduling*. PhD thesis, Department of Mathematics, Technische Universität Berlin, Germany, 1996.

[20] W.E. Smith. Various optimizers for single stage production. *Naval Research Logistics*, 3(1):59–66, 1956.

[21] J.P. Sousa and L.A. Wolsey. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, 54(3):353–367, 1992.

[22] W.A. Wallace, D. Mendonca, E. Lee, J.E. Mitchell, and J. Chow. Managing disruptions to critical interdependent infrastructures in the context of the 2001 World Trade Center attack. *Beyond September 11th: An Account of Post-Disaster Research*, pages 165–198, 2003.

[23] H. Waterer, E.L. Johnson, P. Nobili, and M.W.P. Savelsbergh. The relation of time indexed formulations of single machine scheduling problems to the node packing problem. *Mathematical Programming*, 93(3):477–494, 2002.

[24] N. Xu, S.D. Guikema, R.A. Davidson, L.K. Nozick, Z. Cagnan, and K. Vaziri. Optimizing scheduling of post-earthquake electric power restoration tasks. *Earthquake Engineering and Structural Dynamics*, 36(2):265–284, 2007.

Appendix

Incorporating the DC Model for Power Infrastructures

The DC model is a linear approximation of the behavior of a power system that is typically used in modeling the behavior of the power infrastructure system, especially the transmission network. For example, Bienstock and Mattia [4] used the DC model in modeling problems related to mitigating grid blackout problems. We note that Bienstock and Mattia [4] also provide a detailed discussion on the relationship between the linear DC model and the nonlinear AC model for a power system. The DC model includes decision variables at each node of the network that represent the *phase angle* of the node. The flow on arc (i, j) is then a function of the phase angles of nodes i and j along with the reactance of the arc (i, j). The reactance, b_{ij}, of the arc is dependent on the length of it and the voltage levels. By defining θ_i for $i \in N$ as the phase angle of node i, the flow on arc (i, j) is determined through the equation:

$$b_{ij}x_{ij} = (\theta_i - \theta_j). \tag{17}$$

We note that both the phase angle variables *and* the arc flow variables are unrestricted in the DC model. A negative flow on arc (i, j) corresponds to power flowing from node j to node i. Therefore, it is necessary to include constraints that model equation (17) into the core INDS problem.

The difficulty, however, is that this constraint should only be enforced at time t for the arcs in the network that have been completed prior to time t. In order to incorporate the DC model into (IP), we will define variables θ_{it} for $i \in N$ and $t = 1, \ldots, T$ that represent the phase angle of node i in time period t. For all arcs $(i, j) \in A$, we replace constraints (5) with the constraints

$$b_{ij}x_{ijt} = (\theta_{it} - \theta_{jt}) \quad \text{for } (i,j) \in A,\ t = 1, \ldots T, \tag{18}$$

$$-u_{ij} \leq x_{ijt} \leq u_{ij} \quad \text{for } (i,j) \in A,\ t = 1, \ldots, T. \tag{19}$$

These constraints enforce the DC flow model for arcs $(i, j) \in A$ and ensure that the flow on the arc does not exceed its capacity (thus preventing failure of these components). For arcs $(i, j) \in A'$, we can model the fact that we only wish to enforce DC flow calculations (17) when arc (i, j) appears in the network at time t by using 'Big-M' constraints. In particular, for arcs $(i, j) \in A'$, we replace constraints (6) with

$$b_{ij}x_{ijt} \leq (\theta_{it} - \theta_{jt}) + M(1 - \beta_{ijt}) \ \text{ for } (i,j) \in A',\ t = 1, \ldots T, \tag{20}$$

$$b_{ij}x_{ijt} \geq (\theta_{it} - \theta_{jt}) - M(1 - \beta_{ijt}) \ \text{ for } (i,j) \in A',\ t = 1, \ldots, T \tag{21}$$

$$-u_{ij}\beta_{ijt} \leq x_{ijt} \leq u_{ij}\beta_{ijt} \quad \text{for } (i,j) \in A',\ t = 1, \ldots, T. \tag{22}$$

If $\beta_{ijt} = 0$, then these constraints force $x_{ijt} = 0$ while not imposing any restrictions on the relationship between the phase angles of nodes i and j due to the big M. If $\beta_{ijt} = 1$, then constraints (20)-(21) guarantee that the DC flow equation (17) is satisfied for arc (i, j) in time period t while constraint (22) guarantees that the capacity of the arc is not violated. We will refer to the core INDS problem where we have replaced constraints (5)-(6) with constraints (18)-(22) as the INDS problem with the DC Model.