Decision Support for Disaster Management Through Hybrid Optimization

by
Carleton James Coffrin
Sc.B., University of Connecticut, 2006
B.F.A., University of Connecticut, 2006
Sc.M., Brown University, 2010

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2012

This dissertation by Carleton James Coffrin is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Pascal Van Hentenryck, Director

Recommended to the Graduate Council

Date _____ _____

Russell Bent, Reader
Los Alamos National Laboratory

Date _____ _____

Claire Mathieu, Reader
Brown University

Date _____ _____

Laurent Michel, Reader
University of Connecticut

Approved by the Graduate Council

Date _____ _____

Peter M. Weber
Dean of the Graduate School

Contents

Abstract of "Decision Support for Disaster Management Through Hybrid Optimization" by Carleton James Coffrin, Ph.D., Brown University, May 2012

Recent natural disasters, such as Hurricane Katrina in 2005 and the Japan earthquake in 2011, have demonstrated that situational awareness, the focus of much research in disaster management, is not sufficient to provide effective mitigation, response, and recovery. The United States government has recognized the need for decision support to enhance the cognitive abilities of decision makers. Developing such tools is particularly challenging because they require modeling complex interdependent infrastructure systems and must operate under aggressive runtime constraints.

This thesis demonstrates that hybrid optimization technology can produce decision support tools for delivery of relief supplies, restoration of the electrical power infrastructure alone, and joint restoration of interdependent natural gas and electrical power infrastructures. These hybrid methods use mixed integer programming for infrastructure modeling, large neighborhood search for routing of repair crews, and randomized adaptive decomposition for scaling the optimization algorithms to practical sizes. The proposed hybrid methods increase the restoration quality significantly over current field practices and outperform traditional technologies, which cannot find feasible solutions within the runtime constraints.

The optimization algorithms must rely on linear approximations of the nonlinear AC power flow equations to meet the runtime constraints of these applications. Unfortunately, the accuracy of the traditional linearized DC model degrades significantly as network disruptions increase. This thesis also proposes novel linear approximations that are shown to provide significant improvements in accuracy under disruptions.

Chapter 1

Introduction

1.1 Motivation

The digital revolution has made many broad changes in modern society, two of them of particular importance to this research. First, the digitization of record keeping has enabled businesses to integrate logistical optimization more than ever before. For example, a state-of-the-art inventory management system is essential to WalMart's business model [57]; online advertising is settled in millions of micro-auctions every second [11]; power prices are set in less-than-five-minute intervals using a unit commitment auction [60]. Second, the networking of many different technologies has created tighter interdependence among infrastructure systems. For example, live traffic information is streamed to drivers to prevent traffic jams; that information is provided via an antenna that is run on the power grid, and the power grid is supplemented by natural gas turbines [72]. It is possible that an extreme temperature drop could disrupt the natural gas distribution system and through this chain of dependencies cause a traffic jam. Furthermore, many of these systems are codependent. For example, communication networks require electricity to operate, while many *smart-grid* technologies require communication to operate properly; electricity-based compressors are needed to move natural gas, while natural gas is used to generate electricity.

The benefits of the digital revolution are many. It has provided near real-time data that was only dreamed of several decades ago and has allowed us to maintain a high quality of life despite shrinking economies. However, all this efficiency has come at the cost of robustness. Integrated inventory-management systems let businesses reduce storage costs, but these systems can be very brittle to unexpected demands or supply-chain disruptions (it is no coincidence that WalMart takes disaster relief very seriously). Vehicle routing software may allow shipping companies to hire fewer daily truck drivers, but the companies are now dependent on an operational communication network in addition to a transportation network. The optimization of operations and the interdependence of infrastructures have a compounding effect on fragility that can increase the negative effects of large disruptions and create *weak links* that cascade into major crises [22]. There is a fundamental

need to understand this increased fragility and provide strategies to mitigate the effects of significant disruptions, both natural and manmade.

Understanding the fragility of a system of interdependent infrastructures is challenging. Many infrastructure behaviors are governed by physical laws, such as Ohm's law in energy systems or the compressible flow equations in natural gas systems. These physical equations are usually nonlinear and require iterative algorithms, such as Newton's method, for evaluation. Other infrastructures, such as the transportation network, are governed by many individual agents (car drivers) who follow a set of ad-hoc behaviors. The most accurate models of these systems simulate each individual agent and consume huge computational resources [20]. Providing decision support for these kinds of infrastructure systems adds another level of complexity, because at the very least the systems must be evaluated under several different operating conditions. For example, one might consider which road should be expanded to prevent a daily traffic jam. Without significant restrictions on drivers' behavior, a naive algorithm may need to try many possible road expansions before arriving at a good solution. Furthermore, providing decision support often involves logistics optimization problems that are NP-hard, such as vehicle routing, warehouse location, and scheduling.

Many of the important considerations regarding infrastructure systems are open frontiers. A few infrastructure considerations of particular interest are: expansion planning, how to increase the capacity of the infrastructure cheaply while ensuring some minimal robustness; vulnerability analyses, identifying the weakest links in a infrastructure, usually assuming an intelligent adversary; and disaster preparation and recovery, mitigating the negative effects of natural and manmade disasters. This work focuses on the last area of interest, disaster management, a selection inspired by the U.S. Department of Homeland Security's identification of the need for decision support tools in this area [99]. Furthermore, seasonal hurricanes pose a regular and predictable threat to the United States infrastructure, so that disaster preparation and recovery tools have an immediate societal impact.

In addition to the challenges already discussed, the disaster management context has two additional properties that make it particularly demanding. The first is that disasters are inherently stochastic. This requires that decision support tools consider many different contingencies and produce solutions that are flexible, with several possible outcomes. The second property is aggressive runtime constraints. Disasters often occur unexpectedly or with only a few days' warning. In either case, decision support tools must suggest possible courses of action within a few minutes or hours — they cannot take days. After an earthquake, for example, the results of a decision support tool are needed as soon as possible because every minute may save lives. And while hurricanes typically arrive with several days' warning, a decision support tool may be needed early on to decide if additional repair crews should be called in from neighboring regions, perhaps several hundred kilometers away. This decision must be made quickly because travel time of the repair crews is significant.

The Department of Homeland Security's prioritization of disaster recovery decision support tools was motived by two factors [99, 105]. First, infrastructure operators traditionally work independently and significant improvements may be made by considering interdependencies [25]. Second, modern

best practices in disaster recovery often rely on human intuition and domain experts. For example, Florida's power engineers at highly skilled are restoring their power grid because of the regular damage caused by seasonal wind storms, while New England's power engineers lack this expertise. A robust algorithmic solution to infrastructure restoration would improve the United States disaster response in two ways. First, it would provide all areas of the country with the same effective restoration practices. Second, replacing human intuition and greedy heuristics with optimization technology can bring significant improvements in the quality of all restoration procedures, including current expert plans.

The U.S. government has invested in the science of disaster preparedness for several decades [4]. However, until recently, these efforts have focused on *situational awareness*: analysis of how various infrastructures will be affected by a projected threat. This awareness is achieved through state-of-the-art physical simulations of both threat behavior and system fragility. For example, when a hurricane is imminent, weather simulation tools generate possible landfall contingencies, each of which is then combined with an infrastructure fragility simulation that evaluates damage effects based on wind speeds. The results of both of these simulations yields merely one damage scenario, but such simulation algorithms are stochastic and can be run several times to generate an ensemble of possible contingencies. Although *situational awareness* is a necessary first step in understanding disaster preparedness, the work presented here goes a step further to provide *decision support*, operational recommendations on how to mitigate the effects of a threat. The decision support tools developed here leverage previous work on situational awareness by performing stochastic optimization over an ensemble of damage contingencies generated with existing situational awareness pipelines.

It is clear that decision support is needed for the increasingly complex infrastructures of the twenty-first century. Such support is computationally challenging. This work aims to demonstrate that

> **Thesis Statement**: Hybrid optimization algorithms can solve challenging disaster management problems and improve current field practices.

Here hybrid optimization algorithms are novel hybridizations of constraint programming, integer programming, local search, and stochastic optimization. This statement is demonstrated by developing a general approach to solving a broad class of disaster recovery problems that is validated on three case studies of increasing difficulty and complexity. The remaining sections of this chapter formalize the disaster recovery problem, present a general solution framework, and discuss related work.

1.2 Formalization

In order to develop a computational model, the disaster recovery problem must be formalized. While any computational model of a real system is necessarily an approximation of the real world, the model presented here attempts to have enough detail to capture current field practices accurately. Two primary aspects must be formalized in the disaster recovery problem: how to model the

Given:

- Infrastructure Networks: $\mathcal{I}$
- Locations: L, $loc()$
- Repositories: $i \in R$
  - Capacity: RC_i
  - Investment Cost: RI_i
  - Maintenance Cost: RM_i
- Vehicles: $i \in V$
  - Capacity: VC
  - Start Location: H_i^+
  - End Location: H_i^-
- Damage Scenarios: $i \in S$
  - Scenario Probability: P_i
  - Damaged Locations: $DL_i \subset L$
  - Location Quantity: $Q_{i,j \in L}$
  - Location Maintenance Time: $M_{i,j \in L}$
  - Travel Time Matrix: $T_{i,j \in L, k \in L}$
- Budget: B

Output:

Components stored at each repository
Repair schedules for each vehicle in each scenario

Minimize:

f(Unserved Demands,
Restoration Time,
Restoration Costs)

Subject To:

Vehicle and repository capacities
Vehicles start and end locations
Costs $\leq B$

Figure 1.1: Specification of the Abstract Disaster Recovery Problem.

infrastructures and how to handle the stochasticity of the disasters.

Most infrastructure systems form a network, suggesting that a graph representation is appropriate. For example, in the transportation system different destination nodes are connected by road edges, while power networks connect generation and load nodes by power lines. Infrastructure systems, such as the storage warehouses, that are not connected by edges can be modeled by a disconnected graph. All these infrastructure graphs can be combined into a supergraph containing all the edges and nodes. Each node in the supergraph is connected through the transportation network.

There are many possible choices for stochastic models. In the context of disaster recovery, explicit disaster scenarios are a natural choice for several reasons. First, they allow different kinds of damage profiles to be considered simultaneously. Second, they have a nice synergy with current disaster simulation tools that often produce several contingencies, such as hurricane spaghetti models. Third, policy makers, who are the primary users of these tools, can easily understand the meaning of each contingency. For these reasons, this work assumes a set of predetermined disaster scenarios defining how the infrastructures have been disrupted. This is convenient from a modeling standpoint because the stochasticity of the model is represented as a two-stage stochastic program with explicit scenarios.

Figure 1.1 summarizes the abstract disaster recovery problem (DRP) described in detail below.

Infrastructures and Locations The infrastructure systems are quite abstract in this formulation and are refined in each case study. The set of infrastructure networks, $\mathcal{I}$, is simply a collection of graphs each representing a particular infrastructure (e.g. power grid, natural gas pipes). In principle, the storage and transportation infrastructures can be modeled as abstract infrastructures. However, because they have a special role in connecting all the infrastructures and damage scenarios, they are described more explicitly in this formulation. The set of locations, L, is the set of all nodes

in the supergraph that is the union of all the infrastructure models. The function $loc()$ maps any infrastructure item to its element in the location set L.

Storage Infrastructure The warehouse infrastructure used for storing components is described by the set of repositories R. Each repository $i \in R$ has a maximum capacity RC_i to store restoration components. It also has a one-time initial cost RI_i (the investment cost) and an incremental cost RM_i for each unit of commodity to be stored. Since policy makers often have budget constraints, the sum of all costs in the system must be less than a budget B.

Transportation Infrastructure The transportation infrastructure used to travel between various locations is described by a road network. By design, road networks form a single connected component (otherwise cars would become trapped). Exploiting this strongly connected property, a road network can be concisely described by the shortest paths between every pair of locations of interest, i.e. a distance matrix T of size $|L| \times |L|$. The distance matrix is assumed to be symmetrical and the shortest-path construction is a metric space. However, it is rarely a Euclidian space due to speed limits and road debris. In each disaster scenario $i \in S$, the transportation network is damaged, necessitating new shortest-path calculations. These effects are captured by a unique distance matrix for each scenario, T_i. The road network is used by a homogeneous fleet of repair crews $i \in V$. Each repair crew has storage capacity VC and unique starting location H_i^+ and ending location H_i^-.

Stochasticity The stochastic parts of problem are described by a collection of damage scenarios S. Each scenario $i \in S$ has an associated probability P_i and specifies a set of damaged locations DL_i. Damaged and undamaged locations may require new components and Q_{ij} indicates the quantity of components needed at that location, while M_{ij} indicates the time required to service the component at location j. As mentioned above, disruptions to the road network are captured in the location-to-location travel time matrix $T_{i,j\in L,k\in L}$.

Objective and Solutions The objective function aims at minimizing three factors: (1) the number of unserved demands, which are often a secondary effect of restoring infrastructure components; (2) the time to restore those demands; (3) the restoration costs, which are often dominated by preparation and storage costs. In this abstract formulation, the precise objective function is abstracted into the function $f()$, but it is refined in each case study. Even without a precise objective function, in general this a multi-objective optimization problem in which different components of the object clash.

A solution to a disaster recovery problem has two parts. First we need the storage decisions, that is, what components should be stored in each repository. In the two-stage stochastic framework these are called the *first-stage* variables. Then for each disaster scenario, we need a restoration schedule for each vehicle that gives step-by-step instructions for where a vehicle should go to pick up repair supplies and what parts of the infrastructure it should repair. These vehicle schedules

are called *second-stage* variables in a two-stage stochastic framework because they occur after the uncertainty has been revealed.

1.2.1 Scenario Generation

This work focuses on providing decision support tools given a collection of predefined scenarios. In the interest of providing context, it is worthwhile to discuss how these scenarios are generated. Los Alamos National Laboratory, among others, is highly skilled in providing *situational awareness*, i.e. what are the negative effects of threats on various infrastructures. This awareness is achieved through state-of-the-art physical simulations of both threat behavior and system fragility. For example, when a hurricane is imminent, weather simulation tools are used to generate possible trajectories. Each trajectory is then combined with a infrastructure fragility simulation that evaluates damage effects based on wind speeds. The results of these two simulations yield a damage scenario. The case of a contaminated water supply involves simulation of an estuary system and a fresh water distribution system followed by a contagion transmission simulation. This work thus leverages Los Alamos National Laboratory's expertise in hazard and fragility simulation to produce state-of-the-art damage scenarios.

1.3 General Solution Framework

Despite the broad abstractions laid out in Section 1.2, our formulation still admits an interesting solution framework that forms the basis for the case studies. This general framework is inspired by the operating behaviors of policy makers, which impact the objective function and the runtime constraints of the disaster management context.

Objective Simplification Recall that the objective function of the abstract disaster recovery problem is a multi-objective function of *Unserved Demands*, *Restoration Time*, and *Restoration Costs*. From a mathematical standpoint, calculating the Pareto frontier of this objective would be the most revealing; however, this is very challenging computationally without putting significant restrictions on the disaster recovery model (for example, efficient algorithms exist for linear programming models [49, 85, 84]). However, for humanitarian reasons policy makers often think of this multi-objective problem as a lexicographic objective with a constraint on costs (a budget). This translates to: given some maximum cost, first satisfy as much demand as possible, then satisfy it as fast as possible, and finally (if the budget has not been exhausted) spend as little money as possible. Formally, this means that the objective function can be refined to:

$$\textbf{Minimize: } f_1(\text{Unserved Demands}), f_2(\text{Restoration Time}), f_3(\text{Restoration Costs})$$

where the function order indicates the lexicographic priority of each objective function. That policy makers behave this way is particularly convenient from a modeling perspective because lexicographic multi-objective functions can be solved by a series of single-objective models.

Runtime Constraints Recall from Section 1.1 that problems in disaster management have aggressive runtime constraints, so that optimal solutions to these problems are out of reach. For example, consider a nonstochastic variant of the problem with only one damage scenario and no complex infrastructures to model. This simplified problem, called the location routing problem (LRP), is well studied. Optimality proofs for LRPs are very difficult and although local search techniques have been successful in scaling LRPs to reasonable sizes, state-of-the-art algorithms can only prove optimality on problems that are an order of magnitude smaller than real-world disaster recovery benchmarks. Since optimality is unattainable for this very simplified version of the disaster recovery problem, there is little hope of achieving optimality in the disaster recovery context. Given the runtime constraints, merely finding a high-quality solution to the disaster recovery problem is challenging.

Abandoning hope of global optimality opens the possibility of many approximate solution approaches. One method of approximation is decoupling a model into independent stages. While this approach may remove the optimal solution, it is key in the proposed disaster recovery solution framework. The first decomposition is motivated by the two-stage nature of the general formulation. Once the first-stage variables (the storage decisions) are fixed, the second-stage variables become $|S|$ independent problems. Decomposing the problem in this way yields two models. The first is a *stochastic storage model* that focuses on minimizing the unserved demands across all scenarios. The second model is a *restoration routing model* that focuses on using the available repair crews to restore the unserved demands as quickly as possible. Routing problems are particularly challenging NP-hard problems, and in this context evaluation of a routing solution requires simulation of an infrastructure system, which can be computationally expensive (e.g. solving a system of nonlinear equations). Embedding such calculations in state-of-the-art routing algorithm is impractical because they require thousands or millions of objective evaluations and assume these calculations are very cheap computationally. Therefore, this work proposes further decomposing the restoration routing algorithm to decouple the infrastructure simulation and routing problems. This process is done in three steps. First, a *restoration set problem* is solved to identify which infrastructure items are critical in meeting all the demands. The restoration set is followed by a *restoration order problem* that ignores travel times and assumes the items are restored in some sequence (i.e. one after another). It then orders the critical restoration components so as to restore the demands as fast as possible. Finally, this order is enforced inside a routing model using precedence constraints. The rationale for this decomposition is discussed in detail in Chapter 3.

Figure 1.2 presents the complete decomposed disaster recovery algorithm. Although this decomposition is unlikely to find an optimal solution, the case studies will demonstrate that it always outperforms current best practices and is reasonably close to some coarse bounds found through problem-specific relaxations. These results are not surprising because this decomposition has several nice properties that let it maintain high-quality solutions. The decomposition of the first- and second-stage variables is natural and has the additional benefits of splitting the problem into a more

```
MULTI-STAGE DRP(DRP P)
1  D ← StochasticStorageProblem(P)
2  for s ∈ S
3  do R ← RestorationSetProblem(P_s, D)
4      O ← RestorationOrderProblem(P_s, R)
5      F_s ← PrecedenceRoutingProblem(P_s, O)
6  return F
```

Figure 1.2: General Framework for Solving Disaster Recovery Problems.

classical stochastic optimization problem followed by more classical vehicle routing problems. Because of this separation, more techniques can be borrowed from the communities that study these topics. Additionally, this separation cleanly splits the objective function: the first stage focuses primarily on minimizing unmet demands and the second stages focus on minimizing the time component. This decomposition significantly improves the algorithm's run time for two reasons. First, the second-stage problems become independent and can be solved in parallel. Second, each problem in the decomposed algorithm can be solved with the most appropriate and effective optimization technique (a variety of orthogonal optimization techniques for these different kinds of problems can be found in the stochastic optimization and vehicle routing literature).

1.3.1 Hurricane Case Study

All experimental results in this work use hurricanes as the threat of interest, for several reasons. First, seasonal hurricanes threaten the U.S. coast every year and significant resources are spent preparing for and recovering from them. Due to the regularity and predicability of hurricanes, the United States government has made significant investments in hurricane simulation and wind-speed fragility analysis, yielding a state-of-the-art simulation technology called HAZUS [4] developed and maintained by the Federal Emergency Management Agency (FEMA) and used by the National Hurricane Center, among others. This technology provides us with highly accurate threat and fragility simulations.

Second, the Department of Homeland Security has developed a *fast-response* center that provides situational awareness and decision support whenever a hurricane above category two is projected to make landfall on the United States. By collaborating with Los Alamos National Laboratory on seasonal hurricane threats, this work has a fast track to real-world application in the *fast-response* center.

While this work has focused its attention on applications of immediate need, careful consideration has been given to making the algorithms independent of the disaster profile. It is likely that these algorithms can be applied to a broad set of threat types, but such an extensive study is outside the scope of this work.

1.3.2 Infrastructure Granularity

Section 1.1 briefly touched on the computational challenges of modeling infrastructure systems, and these are now discussed in detail. Infrastructure systems such as power networks, natural gas distribution systems, and water distribution systems are governed by physical laws that manifest as systems of nonlinear equations. Even these nonlinear systems are just an approximation of the real-world behavior of the system, and different levels of model granularity may be achieved through different physical assumptions. For example, consider a Newtonian or an Einsteinian model of the world. While both are approximations of the real world, the latter would yield a more accurate and complex system of equations. In fact, there are many levels of granularity to consider when modeling an infrastructure system. The models fall into two major categories, transient models that take into account the system fluctuations over time, and steady-state models that capture the system's state at one particular instant of time. This work focuses on steady-state infrastructure models.

Within the class of steady-state models are several choices for each infrastructure system. Models of power networks will be used as examples because they are one of the most well studied infrastructure systems. The most popular power flow model is the single-phase AC power flow equations. This model is nonlinear because it involves the product of two free variables and is traditionally solved using Newton's method. However, convergence of Newton's method is not guaranteed for this model and the network's configuration often needs to be modified to assure solution convergence. The next most popular model is the linearized DC power flow model, which is a linear potential flow. This model is advantageous because it is a system of leaner equations and can be solved using linear equation solvers or linear programming. However, this model ignores the reactive power considerations and may have inaccurate line loads as well as voltage regulation problems. Another linear approximation is a maximum-flow model. This model does not accurately model the power's flow through the network, but still can take the line capacities under consideration. Lastly, a connectivity analysis is a very coarse model of power flow that can be used to check if enough supply exists within a connected component to serve all the loads. This model is very fast to compute, but ignores all capacity constraints and is very optimistic. Despite their coarseness, the last two models have been used in industrial and civil engineering studies of power girds.

This work focuses on the most accurate linear infrastructure models. A linear model is advantageous for several reasons. First, it is computationally tractable and does not require iterative methods, and hence solutions are much easier to obtain. Linear models can also be embedded into existing optimization technologies, such as linear programming, which makes them very appealing for decision support tools. In the power engineering community the linearized DC power flow model has been widely accepted for decision support tools despite its potential inaccuracies.

1.4 Research Approach

The most general preparation and disaster recovery problem (Figure 1.1) can be characterized as a two-stage stochastic warehouse location problem with embedded interdependent nonlinear systems.

Solving such a complex problem is a significant challenge and requires leveraging techniques from many areas and inventing novel ways of combining them. Instead of jumping directly to the most complex variant of this disaster preparation and recovery problem, the task can be made more manageable by starting with a simple variant of the problem and progressively increasing its difficulty. For that reason, this work has been broken into three case studies of the disaster preparation and recovery problem of increasing complexity.

The first case study, *Distribution of Relief Supplies* (Chapter 2), focuses on how to deliver relief supplies to shelters after an evacuation has occurred. This case study adds one simple infrastructure system, the relief shelters. Relief shelters are not networked together and are easily modeled, which allows this case study to focus on the non-infrastructure aspects of the problem. This case study is also the variant of the problem closest to the existing literature: it can be thought of as a stochastic variant of the warehouse location problem. The second case study, *Restoration of the Power Network* (Chapter 3), focuses on restoring one complex infrastructure system. The third and final case study, *Restoration of the Natural Gas and Power Networks* (Chapter 4), extends the work in Chapter 3 to consider restoration of two interdependent infrastructures. These three case studies will demonstrate that the generic solution framework presented in Figure 1.2 is a flexible and robust approach to solving disaster recovery problems and can meet the runtime and quality constraints of disaster management applications.

1.5 Related Work

This work spans many research areas including stochastic optimization, multi-objective optimization, combinatorial optimization, humanitarian logistics, and power system restoration, to name a few. This sections reviews a number of components of the problem that are common to each case study. Within each case study additional related work is highlighted to give additional context to that chapter. In particular, infrastructure-specific related work are held until that infrastructure is first introduced.

1.5.1 Humanitarian Logistics and Disaster Management

The operations research community has been investigating the field of humanitarian logistics since the 1990s, but recent disasters have brought increased attention to these kinds of logistical problems [106, 12, 99, 47]. The wide variety of optimization problems in humanitarian logistics combine aspects from classic problems in inventory routing, supply chain management, warehouse location, and vehicle routing. However, the problems posed in humanitarian logistics add significant complexity to their classical variants, and the operations research community recognizes the need for novel research in this area [106, 12]. Some of the key features that characterize these problems are:

1. **Multi-Objective Functions** — High-stake disaster situations often must balance conflicting objective goals (e.g. operational costs, speed of service, and unserved demands) [10, 42, 8, 53].

2. **Non-Standard Objective Functions** — A makespan time objective in VRPs [10, 23] or equitability objectives [8, 23, 63].

3. **Arbitrary Side Constraints** — Limited resources, fixed vehicle fleet [8], fixed latest delivery time [10, 8], or insufficient preparation budget [42, 51].

4. **Stochastic Aspects** — Disasters are inherently unpredictable. Preparations and recovery plans must be robust with respect to many scenarios [42, 53].

Humanitarian logistics also studies these problems at a variety of scales in both space and time. Some problems consider a global scale with time measured in days and weeks [42], while others focus on the minute-by-minute details of delivering supplies from local warehouses directly to the survivors [8, 10, 52]. This work considers a scale often called the *last mile* of distribution and involves warehouse selection and delivery tasks at the city and state scale.

The operations research community has formulated these problems mainly using MIP models. Many humanitarian logistics problems are complex and MIP formulations do not always scale to real-world instances [8, 10, 52]. Additionally, it has been shown that MIP solvers can have difficulty with some of the unique features of these kinds of problems even when problem sizes are small (e.g., minimizing the latest delivery time in VRPs [23]). Local search techniques are often used to scale the problems to real-world instances [10, 23, 52]. This work demonstrates how problem decomposition methods can yield high-quality solutions to such challenges. To the best of my knowledge, this work is the first humanitarian logistic problem to investigate the "last mile" vehicle routing problem and stochastic disaster information simultaneously.

1.5.2 Multi-Objective Optimization

Multi-objective optimization, also called goal programming in the operations research community, has been studied since at least the late 1970s [64]. In the case of a linear model, such as linear programming or integer programming, a multi-objective problem may be expressed as a linear combination of the different objective functions. This is advantageous because several non-dominated solutions may be obtained by giving different weights to each objective function. This is of particular interest for the disaster preparation and recovery problem because the stochastic storage stage often manifests as a mixed integer program and can support a weighted multi-objective scheme. However, each objective function may not be in comparable units, and selection of proper weights may be difficult. In the case of a linear multi-objective function, there also exist algorithms for enumerating a subset of the Pareto frontier [49, 85, 84]. It is likely that these algorithms can be integrated into this work to provide a collection of non-dominated disaster recovery solutions.

In the case of a lexicographic multi-objective function, the goal-programming community has developed a general framework for solving the problem as a series of single-objective problems [64]. The sub-objectives are considered in descending importance and, at each step, one sub-objective is optimized in isolation and side constraints are added to enforce the optimization of the previous steps. This is similar to the decomposition approach used in this work.

1.5.3 Optimization Paradigms

A central theme of this work is that hybridizing different optimization paradigms producing algorithms that are more robust than any one individual optimization technique. Hence, the focus is on combining existing optimization technologies in novel ways. This section reviews several existing optimization technologies that are the building blocks for the rest of this work.

Optimization technologies are classified into two broad categories, complete search and local search. Complete search algorithms guarantee that with enough time they will find the best possible solution (i.e. the *global optimum*). Local search algorithms provide no such guarantee and often produce only a *locally optimum* solution. Due to the optimality guarantee, complete search methods tend to be slow and systematic. In contrast, local search approaches tend to be fast and stochastic. The complete and local search algorithms exhibit a tradeoff between performance and quality that is fundamental and unavoidable. One of the core ideas of this work is to use a combination of both complete and local search algorithms to achieve the high-quality of complete search within the runtime constraints of the disaster management applications.

Three of the most common complete search paradigms are, linear programming (LP) [37], mixed integer programming (MIP) [27], and constraint programming (CP) [58]. Unlike MIP and CP, which can model any NP-hard problem, LPs have significant limitations. All of the decision variables must be continuous and the constraints may only be linear inequalities of those variables. Due to these limitations, LPs solve quickly and guarantee global optimality. MIPs have similar restrictions on their constraints as LPs, but they allow variables to take on discrete values. This subtle change makes MIPs NP-hard to solve in the general case. Due to their close connection to LPs, MIPs are often solved by a *branch-and-bound* approach: a search tree is constructed by *branching* on the discrete variables and the quality of each branch is *bounded* by an LP relaxation of the MIP model. Many variants for solving MIP models exist, such as column-generation, branch-and-price, and Bender's decomposition; a review of these techniques is outside the scope of this document. CP is an orthogonal approach to complete search that is designed to reason on discrete variables. At its core, the CP solver uses a *branch-and-prune* algorithm to solve optimization problems. A search tree is constructed by *branching* on the variables and the combinatorial properties of the constraints are used to *prune* away infeasible branches. CP has proven to be a better optimization tool than MIP when the model constraints have a combinatorial structure that is not easily captured by linear inequalities (e.g. the all-different constraint). In this work, we select the optimization technology (i.e. MIP or CP) that is best suited for the problem at hand.

Unlike complete search, which is characterized by a few paradigms, local search algorithms are too numerous to count. Common local search paradigms include simulated annealing, tabu search, genetic algorithms, constraint-based local search, large neighborhood search, variable neighborhood search, and randomized adaptive decomposition, to name a few. A comprehensive review of most of these approaches can be found in [45]. The present work primarily uses two local search paradigms large neighborhood search (LNS) [46] and randomized adaptive decomposition (RAD) [16, 17, 81]. When one would prefer a complete search algorithm such as MIP or CP but scalability or run time

limits are a problem, LNS is a natural choice for overcoming these limitations. By design, LNS uses a complete search algorithm to explore a large combinatorial neighborhood efficiently. This increases the performance and scalability of a MIP and CP models but sacrifices their global optimality guarantee. For very challenging problems, LNS may not provide sufficient scalability. In such cases, problem decompositions provide another level of scalability. The RAD is a robust framework for combining LNS and problem-specific decompositions to achieve another level of scalability. The LNS and RAD approaches are preferred in this work as they exploit the strengths of complete search algorithms, such as MIP and CP, but provide the scalability and performance required in disaster management applications.

1.6 Contributions

Since this thesis builds heavily on existing optimization techniques, it is important to highlight where the novel contributions of this work begin. The primary contribution of this work is the general solution framework for solving disaster recovery problems, Figure 1.2. In hindsight, this framework may seem natural and even obvious but throughly demonstrating its practicality on a number of application domains is a significant task. Throughout each chapter a detailed implementation of the general framework is presented. And many novel optimization models and algorithms are developed and evaluated. Each of these algorithms may be characterized as a MIP, LNS, or RAD approach, but the detailed implementations of each general optimization technique and their interconnections are novel. In fact, this work presents only those optimization algorithms that where found effective, although many others were explored. The detailed technical contributions are discussed in each chapter, but every model, decomposition, and algorithm presented in this thesis is a novel contribution.

Chapter 2

Distribution of Relief Supplies

The distribution of relief supplies in the first few days after a major disaster is critical to human welfare. Dehydration and exposure are serious concerns at relief shelters. This chapter solves the following abstract disaster recovery problem: how to store a single commodity throughout a populated area to minimize its delivery time after a disaster has occurred. This case study has only one simple infrastructure system, which is the relief shelters. Because these shelters are not networked together they can be modeled easily and this problem reduces to the simplest variant of the abstract disaster recovery problem formulated in Section 1.2. In fact, this problem will not need the restoration set problem or the restoration order problem stages, but will develop a highly specialized version of the precedence routing problem. Despite its simplicity in the disaster recovery framework, this study makes the following technical contributions:

1. It formalizes the single-commodity allocation problem (SCAP).
2. It proposes a multi-stage optimization framework for solving SCAPs.
3. It proposes several modeling enhancements to improve solution quality and scalability.
4. It demonstrates how the solution framework can be used to provide decision support.
5. It validates the approach on the delivery of potable water for hurricane recovery, using disaster scenarios generated by state-of-the-art hurricane simulation tools similar to those used by the National Hurricane Center.
6. It demonstrates that the solution framework brings significant benefits over current field practices.

The rest of this chapter is organized as follows. Section 2.1 presents a mathematical formulation of the disaster recovery problem and sets up the notation for the rest of this chapter. Section 2.2 presents the solution framework using mathematical models. Section 2.3 presents a number of modeling and algorithmic improvements that refine each of the initial models and allow them to

Given:
- Locations: $L = R \cup H$, $loc()$
- Repositories: $i \in R$
  - Capacity: RC_i
  - Investment Cost: RI_i
  - Maintenance Cost: RM_i
- Vehicles: $i \in V$
  - Capacity: VC
  - Start Depot: H_i^+
  - End Depot: H_i^-
- Scenario Data: $i \in S$
  - Scenario Probability: P_i
  - Damaged Locations: $DL_i \subset L$
  - Location Quantity: $Q_{i,j \in L}$
  - Travel Time Matrix: $T_{i,j \in L, k \in L}$
- Budget: B

Output:
- Quantity of commodities stored at each warehouse
- Delivery schedules for each vehicle in each scenario

Minimize:
- Unserved Demands,
- $\text{MAX}_{i \in V}$ Tour Time$_i$,
- Storage Costs

Subject To:
- Vehicle and site capacities
- Vehicles start and end locations
- Storage Costs $\leq B$

Notes:
- Every warehouse that stores commodities must be visited at least once

Figure 2.1: The Abstract Disaster Recovery Problem Specification Specalized for the Distribution of Relief Supplies.

scale to very large instance sizes. Section 2.4 investigates whether the problem decomposition yields a significant reduction in solution quality. Section 2.5 discusses challenges and solutions for the vehicle routing aspects of the problem. Section 2.6 takes a step back and demonstrates how the solution approach is used in practice to provide decision support. Section 2.7 validates the approach experimentally and Section 2.8 concludes the chapter.

2.1 The Single-Commodity Allocation Problem

The SCAP begins with the abstract formation in Figure 1.1 and makes a number of specializations. The first specialization is the integration of the relief shelter infrastructure. Relief shelters are locations to which people flee after a disaster. The amount of commodities a shelter needs is proportional to the number of people who flee to that location. Shelters additionally have the ability to store some commodities at their location. This means that shelters are always co-located with a repository. Additionally, the demands for commodities at any shelter are captured by the damage scenarios. In fact, the relief shelter infrastructure is so simple that it is modeled implicitly by the repositories and the damage scenarios, which capture the storage and demands respectively. Figure 2.1 summarizes the entire SCAP specification, which we now describe in detail.

Objectives The objective function aims at minimizing three factors: (1) the amount of unsatisfied demands; (2) the time it takes to meet those demands; (3) the cost of storing the commodity. As discussed in Section 1.3, this work will assume a lexicographic objective; however, the stochastic storage model in this application is a MIP and thus can easily support a linear combination of the objectives. The routing objective in this application is to minimize the time of the last delivery. Although other objectives are worth consideration, this one was selected because it is preferred

by the U.S. Department of Homeland Security. Minimizing the time of the last delivery is a very difficult aspect of this problem, as demonstrated in [23].

Side Constraints Each repository $i \in R$ has a maximum capacity RC_i to store the commodity. It also has a one-time initial cost RI_i (the investment cost) and an incremental cost RM_i for each unit of commodity to be stored. As policy makers often work within budget constraints, the sum of all costs in the system must be less than a budget B. Every repository can act as a warehouse and a customer, and its role changes on a scenario-by-scenario basis depending on site availability and demands. Additionally, if a repository is acting as a warehouse for its own demands, a vehicle must still visit that location before the stored commodities are available for consumption (for example, a government official must unlock the storage room).

SCAPs also feature a fleet of V vehicles that are homogeneous in capacity VC. Each vehicle $i \in V$ has a unique starting depot H_i^+ and ending depot H_i^-. Unlike classical vehicle routing problems [97], customer demands in SCAPs often exceed the vehicle capacity and hence multiple deliveries are often required to serve a single customer.

Stochasticity SCAPs are specified by a set of S different disaster scenarios. Scenario $i \in S$ has an associated probability P_i and specifies the set DL_i of sites that were damaged during the disaster. Moreover, scenario i specifies, for each repository $j \in R$, the demand D_{ij} and site-to-site travel times $T_{i,L,L}$ that capture the damage to the transportation infrastructure.

Unique Features Although different aspects of this problem have been studied before in the context of vehicle routing, location routing, inventory management, and humanitarian logistics, SCAPs present unique features. Earlier work in location-routing problems (LRP) assumes that (1) customers and warehouses (storage locations) are disjoint sets; (2) the number of warehouses is $\approx$ 3..20; (3) customer demands are less than the vehicle capacity; (4) customer demands are atomic.

None of these assumptions hold in the SCAP context. In a SCAP, it may not only be necessary to serve a customer with multiple trips but, due to the storage capacity constraints, those trips may need to come from different warehouses. The key features of SCAP are: (1) each site can be a warehouse and/or customer; (2) one warehouse may have to make many trips to a single customer; (3) one customer may be served by many warehouses; (4) the number of available vehicles is fixed; (5) vehicles start and end in different depots; (6) the delivery-time objective is to minimize the time of the last delivery.

2.2 The Solution Framework

This section presents the solution framework for the SCAP problem. Variants on this approach are presented in Section 2.3. In keeping with the design requirements discussed in Chapter 1, our goal is to provide a solution framework that is a robust and modular tool in aiding policy makers' decisions about disaster management. It is very important for the tool to run in only a few seconds

or minutes, so that policy makers can make decisions quickly in the aftermath of a disaster and have the opportunity to consider multiple alternative courses of action.

Section 1.3 discussed the computational difficulties of solving general preparation and disaster recovery problems. However, because this variant is one of the simplest variants of the problem, these difficulties are worth considering again. The SCAP problem is a stochastic generalization of the location routing problem. Previous work on location routing (e.g., [98, 5, 78]) has shown that reasoning over the storage problem and the routing problem simultaneously is extremely hard computationally. The additional unique features that make SCAPs more challenging than location routing problems were discussed in Section 2.1. Additionally, a pure MIP formulation of this problem would require approximately 4,000..1,750,000 0/1 decision variables depending on the instance size, outside the range of industrial integer programming solvers. Further exacerbating the challenges is the difficulty of MIP solvers in minimizing the time of the last delivery, as demonstrated in [23].

To address these difficulties and to make the problem tractable under the aggressive disaster recovery runtime constraints, we adopt the general solution framework presented in Section 1.3. However, due to the simplicity of the relief shelter infrastructure, the *RestorationSetProblem* and *RestorationOrderProblem* are not necessary. This is because the *StochasticStorageProblem* can model the *RestorationSetProblem* exactly and all the relief deliveries are considered equally important, making the *RestorationOrderProblem* irrelevant. Both of these features are atypical of more complex infrastructure systems. However, we exploit these properties by making a sophisticated routing algorithm that has three stages, *Customer Allocation*, *Repository Routing*, and *Fleet Routing*. The final SCAP algorithm with *RestorationSetProblem* and *RestorationOrderProblem* stages omitted is a four-stage algorithm that decomposes the storage and routing decisions. The four stages and the key decisions of each stage are as follows:

1. **Stochastic Storage**: Which repositories should store the commodity and how much is stored at each location?
2. **Customer Allocation**: How should stored commodities be allocated to each customer?
3. **Repository Routing**: For each repository, what is the best customer distribution plan?
4. **Fleet Routing**: How can the repositories be visited so as to minimize the time of the last delivery?

The high-level algorithm is presented in Figure 2.2 and each step is discussed in detail throughout this section. Our decomposition assumes the decisions of each stage are independent. Although this decomposition may prevent us from finding a globally optimal solution, the runtime benefits are critical for making this tool usable in practice. Furthermore, previous work has shown that problem decomposition can bring significant runtime benefits with minimal degradation in solution quality [29].

```
MULTI-STAGE-SCAP(SCAP 𝒢)
1  𝒟 ← StochasticStorageProblem(𝒢)
2  for s ∈ S
3  do 𝒞 ← CustomerAllocationProblem(𝒢_s, 𝒟)
4     for w ∈ R
5     do 𝒯 ← RepositoryRoutingProblem(𝒢_s, 𝒞_w)
6     ℱ_s ← FleetRoutingProblem(𝒢_s, 𝒯)
7  return ℱ
```

Figure 2.2: The Decomposition Framework for Solving SCAPs.

2.2.1 Stochastic Storage

The first stage captures the cost and demand objectives precisely but approximates the routing aspects. In particular, the model considers only the time to move the commodity from the repository to a customer, not the maximum delivery times. Let J_s be a set of delivery triples of the form $\langle source, destination, quantity \rangle$ for a scenario s. At the high level, the delivery-time component of the objective can be thought of as being replaced by

$$W_y \sum_{\langle f,t,q \rangle \in J_s} T_{sft} \frac{q}{VC}$$

Model 1 presents the stochastic MIP model, which scales well with the number of disaster scenarios because the number of integer variables depends only on the number of sites n. The meaning of the decision variables is explained in the figure. The main outputs are the number of units $Stored_i$ stored at each location i and the number of units $Sent_{sij}$ sent from location i to location j in scenario s. The objective function is described in lines (M1.1–M1.3): line (M1.1) describes the cost of unsatisfied demand, line (M1.2) the transportation cost, and line (M1.3) the cost of opening a location and storing the commodities. Constraint (M1.4) specifies that the facility costs cannot exceed the budget, constraints (M1.5) specify that a facility must be open to store commodities, and constraints (M1.6) capture the commodity demands. Constraints (M1.7) specify that a location cannot use more commodity than it has in storage. Constraints (M1.8–M1.9) relate the commodity flows and the number of trips between different locations for specific scenarios. Constraints (M1.10) concern facilities destroyed by the disaster that use the stored commodities.

Once the storage decisions are fixed, the uncertainty is revealed and the second stage reduces to a deterministic multi-depot, multi-vehicle capacitated routing problem whose objective consists in minimizing the latest delivery. To our knowledge, this problem has not been studied before. One of the difficulties in this setting is that customer demand is typically much larger than vehicle capacity. As a result, each customer may be served by multiple vehicle trips, and due to warehouse capacity constraints those trips may come from different warehouses. We tackle this problem in three steps. We first decide how the commodities of each warehouse should be allocated to the customers (Customer Allocation). Then we consider each repository independently and determine a number of vehicle trips to serve the repository customers (Repository Routing). A trip is a tour

Model 1 Stochastic Storage and Customer Selection (SSM).

Variables:

$Stored_i \in (0, RC_i)$ – quantity stored at repository i
$Open_i \in \{0, 1\}$ – nonzero storage at repository i
Second-stage variables for each scenario s:
$Outgoing_{si} \in (0, RC_i)$ – total units shipped from repository i
$Incoming_{si} \in (0, Q_{si})$ – total units coming to repository i
$Unsatisfied_{si} \in (0, Q_{si})$ – demand not satisfied at repository i
$Sent_{sij} \in (0, \min(RC_i, Q_{sj}))$ – units shipped from repository i to repository j

Minimize:

$$\sum_{s \in S} P_s \sum_{i \in R} Unsatisfied_{si}, \tag{M1.1}$$

$$\sum_{s \in S} P_s \sum_{i,j \in R} T_{sij}\ Sent_{sij}/VC, \tag{M1.2}$$

$$\sum_{i \in R} RI_i\ Open_i + RM_i\ Stored_i \tag{M1.3}$$

Subject To:

$$\sum_{i \in R} (RI_i\ Open_i + RM_i\ Stored_i) \leq B \tag{M1.4}$$

$$RC_i\ Open_i \geq Stored_i \qquad \forall i \in R \tag{M1.5}$$

$$Incoming_{si} + Unsatisfied_{si} = Q_{si} \qquad \forall s \in S\ \forall i \in R \tag{M1.6}$$

$$Outgoing_{si} \leq Stored_i \qquad \forall s \in S\ \forall i \in R \tag{M1.7}$$

$$\sum_{j \in R} Sent_{sij} = Outgoing_{si} \qquad \forall s \in S\ \forall i \in R \tag{M1.8}$$

$$\sum_{j \in R} Sent_{sji} = Incoming_{si} \qquad \forall s \in S\ \forall i \in R \tag{M1.9}$$

$$Outgoing_{si} = 0 \qquad \forall s \in S\ \forall i \in DL_s \tag{M1.10}$$

that starts at the depot, visits customers, returns to the depot, and satisfies the vehicle capacity constraints. We then determine how to route the vehicles to perform all the trips and minimize the latest delivery time (Fleet Routing).

2.2.2 Customer Allocation

The assignment of customers to repositories is a very important step in this algorithm because it directly affects the quality of the trips computed by the repository routing. Recall that Section 2.2.1 uses

$$W_y \sum_{\langle f,t,q \rangle \in J_s} T_{sft} \frac{q}{VC}$$

as an approximation of travel distance. Notice that this approximation is good when $q \gg VC$, but inaccurate when $q < VC$. Our experimental results indicate that $q < VC$ often occurs with large budgets B, and in that case this approximation yields poor customer allocation decisions. Because the customer allocation problem occurs after the uncertainty of the scenario is revealed, we can solve

Model 2 Customer Allocation of Scenario s (CA).

Variables:

$Sent_{ij} \in (0, Stored_i)$ – units moved from repository i to repository j

$Trips_{ij} \in \{0, 1, \ldots, \lceil Stored_i/VC \rceil\}$ – trips needed from repository i to repository j

Minimize:

$$\sum_{i \in R} (Q_{si} - \sum_{j \in R} Sent_{ji}), \qquad \text{(M2.1)}$$

$$\sum_{i,j \in R} T_{sij} Trips_{ij} \qquad \text{(M2.2)}$$

Subject To:

$$\sum_{j \in R} Sent_{ij} \leq Stored_i \quad \forall i \in R \qquad \text{(M2.3)}$$

$$\sum_{j \in R} Sent_{ji} \leq Q_{si} \quad \forall i \in R \qquad \text{(M2.4)}$$

$$Sent_{ji} = 0 \quad \forall i \in R\, \forall j \in DL_s \qquad \text{(M2.5)}$$

$$Trip_{ij} \geq Sent_{ij}/VC \quad \forall i \in R\, \forall j \in R \qquad \text{(M2.6)}$$

a slightly stronger approximation of the travel distance, specifically

$$W_y \sum_{\langle f,t,q \rangle \in J_s} T_{sft} \left\lceil \frac{q}{VC} \right\rceil$$

which is robust for all values of q. The customer allocation model with the improved travel time objective is presented in Model 2. The objective function is described in lines (M2.1–M2.2): line (M2.1) represents the cost of unsatisfied demand and line (M2.2) represents the cost of transportation. Because the storage decisions have already been made, storage costs do not enter this objective function. Constraints (M2.3) specify that a location cannot use more commodity than it has in storage. Constraints (M2.4) capture the commodity demands. Constraints (M2.5) concerns facilities destroyed by the disaster, and constraints (M2.6) relate the commodity flows and the number of trips between different locations.

This problem must be solved quickly since it is now considered after the uncertainty is revealed, but it is deterministic and tailored to the scenario. Unfortunately, even this simplified problem can be time consuming to solve optimally. However, a time limit of n seconds (where n is the number of repositories) results in solutions within 1% (on average) of the best solution found in one hour. Our results indicate that even suboptimal solutions to this problem yield better customer allocations than those produced by the travel-time approximation in the stochastic storage model.

2.2.3 Repository Routing

Figure 2.3 shows how to create the inputs for repository routing from the outputs of the customer allocation model. For a given scenario s, the idea is to compute the customers of each repository w, the number of full-capacity trips $FullTrips_{swc}$ and the remaining demand $Demand_{swc}$ needed to serve each such customer c. The full trips are considered only in the fleet routing since they must be

For each scenario $s \in S$ *and repository* $w \in R$
$Customers_{sw} = \{i \in R \mid Sent_{swi} > 0, i \neq w\}$
For $c \in Customers_{sw}$
$FullTrips_{swc} = \lfloor Sent_{swc}/VC \rfloor$
$Demand_{swc} = VC(Sent_{swc}/VC - \lfloor Sent_{swc}/VC \rfloor)$
$MinTrips_{sw} = binPack(\{Demand_{swc} \mid c \in Customers_{sw}\}, VC)$

Figure 2.3: The Inputs for the Repository Routing.

Model 3 Repository Routing (RR).

Let:
$N^c_{sw} = Customers_{sw}$ – node customer set
$N_{sw} = \{w\} \cup N^c_{sw}$ – node customers and warehouse set
Variables:
$Edge_{ij} \in \{0, 1\}$ – vehicle travels from node i to node j
$Load_i \in (0, VC)$ – remaining capacity when the vehicle arrives at node i
Minimize:

$$\sum_{i,j \in N_{sw}} T_{sij} Edge_{ij} \tag{M3.1}$$

Subject To:

$$\sum_{j \in N^c_{sw}} Edge_{wj} = MinTrips_{sw} \tag{M3.2}$$

$$\sum_{j \in N_{sw}, i \neq j} Edge_{ij} = 1 \qquad \forall i \in N^c_{sw} \tag{M3.3}$$

$$\sum_{j \in N_{sw}, i \neq j} Edge_{ij} - \sum_{j \in N_{sw}, i \neq j} Edge_{ji} = 0 \qquad \forall i \in N_{sw} \tag{M3.4}$$

$$Load_i \geq Demand_{swi} Edge_{wi} \qquad \forall i \in N^c_{sw} \tag{M3.5}$$

$$Load_j - Load_i \geq Demand_{swi} + M(1 - Edge_{ij}) \qquad \forall i \in N^c_{sw} \ \ \forall j \in N^c_{sw} \tag{M3.6}$$

performed by a round trip. The minimum number of trips required to serve the remaining customers is also computed using a bin-packing algorithm and is solved optimally with a simple and effective tree search.

The repository routing then finds a set of trips serving customers of a repository with minimal travel time. The repository routing is solved using the simple model depicted in Model 3. Line (M3.1) describes the objective that minimizes the sum of the travel times. Constraint (M3.2) fixes the number of trips departing the warehouse. Constraints (3) ensure that each customer is visited by a trip. Constraints (M3.4) are flow-balance constraints that ensure all trips form a cycle and return to the warehouse. Constraints (M3.5–M3.6) make sure each trip does not exceed the vehicle capacity constraint and (M3.6) also ensures that there are no subtours that do not pass through the warehouse. Once all repository routing models are solved, the problem reduces to an uncapacitated multi-depot, multi-vehicle service-time routing problem with service times whose objective consists in minimizing the latest delivery. The fleet routing stage addresses that problem.

Given scenario $s \in S$ *and for each repository* $w \in R$
$Tasks_{sw} = \{t_1, t_2, ..., t_{MinTrips_{sw}}\} \cup FullTrips_{sw}$
For each $t \in FullTrips_{sw}$ to customer c
$Loc_t = w$
$TripTime_t = 2\ T_{swc}$
Let G be the graph formed by the edge set $Edge_{ij} = 1$
Assign each cycle in G a unique task number in $1...MinTrips_{sw}$
For $t \in Tasks_{sw} \setminus FullTrips_{sw}$
$Loc_t = w$
$TaskEdges_t =$ the edges from the cycle in G assigned task number t
$$TripTime_t = \sum_{\langle i,j \rangle \in TaskEdges_t} T_{sij}$$

Figure 2.4: The Inputs for the Fleet Routing.

2.2.4 Fleet Routing

At this stage, the problem reduces to assigning trips to vehicles. Indeed, each trip satisfies the capacity constraints and a vehicle is needed to distribute the commodity to the trip customers. As a result, a trip can be abstracted into a task with a specific location (i.e., its associated repository) and a service time (i.e., the time to perform the trip). The objective is to minimize the latest delivery times. The delivery times take into account both the service times of the tasks and the travel times between repositories. Recall that each vehicle has a starting depot and an ending depot.

Figure 2.4 depicts how to compute the inputs for fleet routing given the results of the earlier steps. The figure shows how to compute the proper service time $TripTime_t$ and location Loc_t for each trip t. The model for the fleet routing is shown in Model 4. This multiple vehicle routing that minimizes the latest delivery time is a straitforward adaptation of the formulation studied in [23] to multiple vehicles. The objective (M4.1) minimizes the maximum arrival time. Constraints (M4.2) expresses that the last arrival time is greater than all other arrival times. Constraints (M4.3) ensure that each task is visited by a vehicle. Constraints (M4.4) are flow balance constraints that ensure the vehicle tours form a cycle. Constraints (M4.5–M4.6) model the vehicle arrival times including the travel and trip service times. The arrival time constraint (M4.6) also ensures that there are no subtours within a vehicle, and constraints (M4.7) make sure each vehicle returns to the correct depot.

2.3 Modeling and Algorithmic Enhancements

We now turn to some modeling and algorithmic improvements to the basic approach that bring significant benefits in real-life applications. The algorithmic improvements are discussed in two groups: first, enhancements to the decomposition that improve solution quality, and second, refinements to the models to scale to very large SCAP instances capturing the state scale.

Model 4 Fleet Routing (FR).

Let:

$Start_s = \{H_1^+, \ldots, H_{|V|}^+\}$ – start nodes
$End_s = \{H_1^-, \ldots, H_{|V|}^-\}$ – end nodes
$N_s^t = \bigcup_{w \in R} Tasks_{s,w}$ – taks nodes
$N_s^+ = Start_s \cup N_s^t$ – departure nodes
$N_s^- = End_s \cup N_s^t$ – returning nodes
$N_s = Start_s \cup End_s \cup N_s^t$ – all nodes
$T_{sij}^f = T_{s,Loc_i,Loc_j} \quad \forall i \in N_s \quad j \in N_s$ – node-based travel-time matrix

Variables:

$Edge_{ijk} \in \{0, 1\}$ – vehicle k travels from node i to node j
$Arrival_{ik} \in (0, \infty)$ – the time vehicle k arrives at node i
$LastArrival \in (0, \infty)$ – the last arrival time of any vehicle

Minimize:

$$LastArrival \tag{M4.1}$$

Subject To:

$$Arrival_{ik} \leq LastArrival \qquad \forall k \in V \;\; \forall i \in N_s \tag{M4.2}$$

$$\sum_{j \in N_s, i \neq j} \sum_{k \in V} Edge_{ijk} = 1 \qquad \forall i \in N_s \tag{M4.3}$$

$$\sum_{j \in N_s, i \neq j} Edge_{ijk} - \sum_{j \in N_s, i \neq j} Edge_{jik} = 0 \qquad \forall k \in V \;\; \forall i \in N_s \tag{M4.4}$$

$$Arrival_{ik} \geq T^f_{sH_k^+ i} Edge_{H_k^+ ik} \qquad \forall k \in V \;\; \forall i \in N_s^- \tag{M4.5}$$

$$Arrival_{jk} - Arrival_{ik} \geq T^f_{sij} + TripTime_i + M(1 - Edge_{ijk}) \quad \forall k \in V \; \forall i \in N_s^t \; \forall j \in N_s^- \tag{M4.6}$$

$$Edge_{H_k^- H_k^+ k} = 1 \qquad \forall k \in V \tag{M4.7}$$

2.3.1 Improving Solution Quality with Path-Based Routing

The delivery plans produced by the proposed approach exhibit an obvious limitation. By definition of a trip, the vehicle returns to the repository at the end of a trip. In the case where the vehicle moves to another repository next, it is more efficient to go directly from its last delivery to the next repository (assuming a metric space, which is the case in practice). To illustrate this point, consider Figure 2.5, where a customer (white node) receives deliveries from multiple repositories (shaded nodes). The figure shows the savings in moving from tour-based (middle) to path-based solution (right). It is not difficult to adapt the algorithm from a tour-based to a path-based routing. In the repository routing, it suffices to ignore the last edge of a trip and to remember where the path ends. In the fleet routing, only the time matrix needs to be modified to take into account the location of the last delivery. To support a path-based fleet routing model, the time matrix must be rebuilt to be indexed by the tasks instead of the repositories.

It is useful to mention one consequence of using the bin-packing algorithm in Figure 2.3. This algorithm aims at reducing the number of tasks in the fleet routing. However, in doing so, it may increase the objective of the repository routing, since additional trips may decrease the overall cost. This is illustrated in Figure 2.6. The leftmost diagram shows the problem specification: the

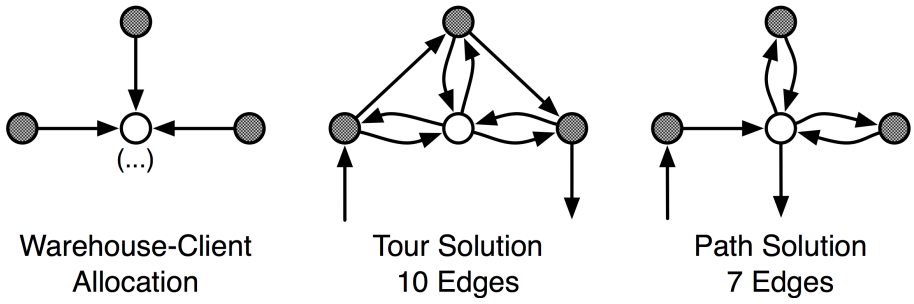

Figure 2.5: Improvement of Path-Based Routing.

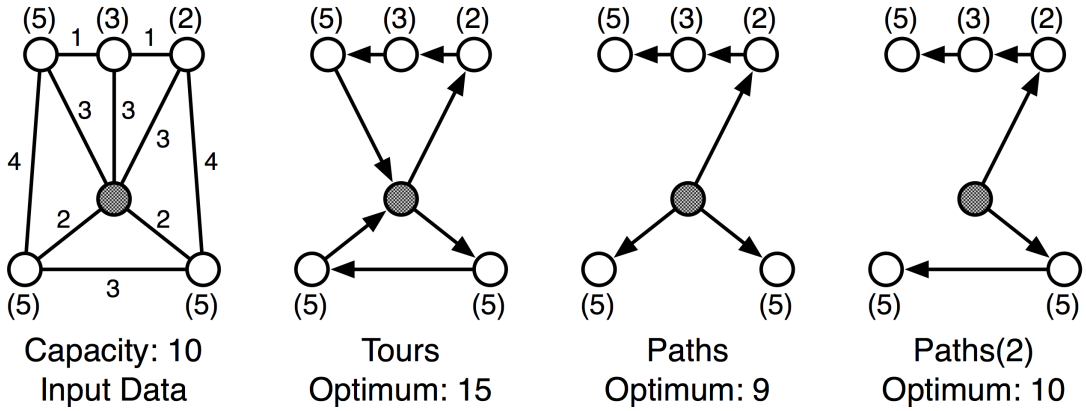

Figure 2.6: Number of Paths Example.

repository, customers, customer demands, vehicle capacity, and travel distances. Recall from the repository routing specification (Model 3) that the objective of this problem is to find the set of tasks with minimum delivery time. The second diagram shows an optimal delivery plan for this problem with tour-based tasks. The third diagram shows an optimal delivery plan with path-based tasks, and the rightmost diagram shows an optimal delivery plan with path-based tasks when the number of tasks is required to be as small as possible. When using path-based reasoning, it is not clear which of the last two diagrams is preferable, but if we consider how each of these approaches affects the results in the fleet routing stage, it seems that fewer full-capacity tasks are preferable to many partial-capacity tasks. Based on this intuition, we make a heuristic choice to require the number of tasks for each repository to be as small as possible. The experimental results indicate that path-based routing brings significant improvements over the tour-based approach.

2.3.2 Scaling the Storage Allocation Model

The runtime results to be presented in Section 2.7 indicate that the Fleet Routing stage of the algorithm is the dominant factor in the algorithm run time for instances with less than 200 storage locations. However, for instances with more than 200 storage locations, the Stochastic Storage Model (SSM) quickly dominates the run time. This is particularly problematic for performance

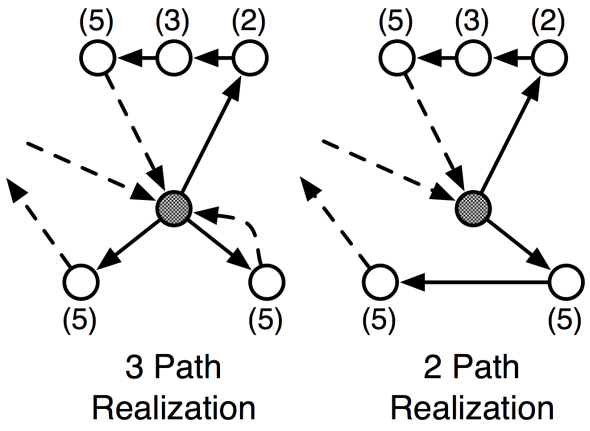

Figure 2.7: Path Routing Realization Example.

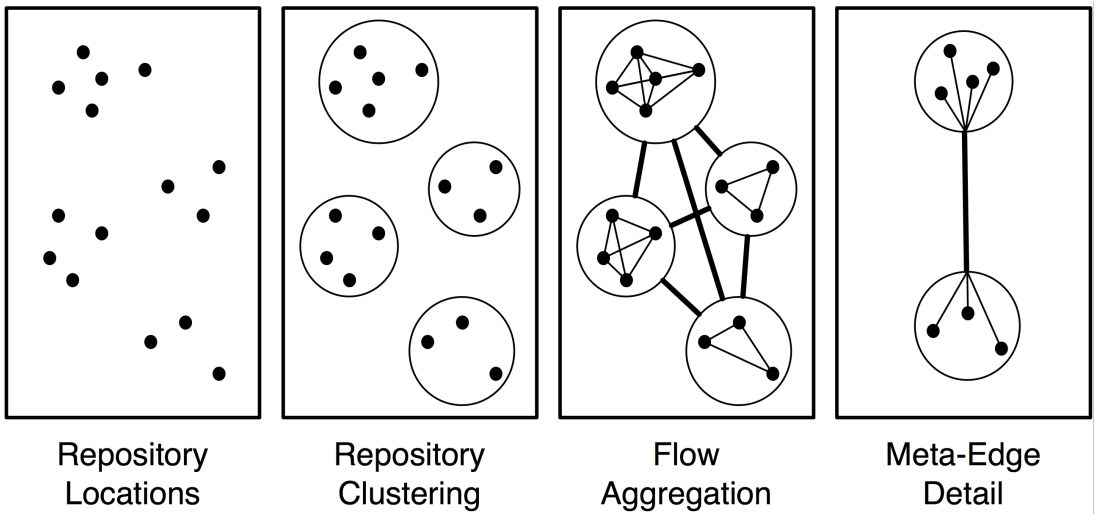

Figure 2.8: Storage Clustering and Flow Aggregation.

because the stochastic storage stage is the only algorithm that cannot be easily parallelized. In this section, we present two alternative models for the stochastic storage problem that provide significant benefits for scalability. Both stochastic storage models rely on a key observation: in the final SCAP algorithm (Figure 2.2), a customer allocation is computed in the SSM and then recomputed in the customer allocation stage. This means when a customer allocation stage is used, only the storage decisions are a necessary output of the SSM. Both of these models achieve faster performance by approximating or ignoring customer allocation in the stochastic storage problem.

Spatial Decomposition of Storage Allocation

In the SSM, the number of variables required for the customer allocation is quadratic in the number of repositories and multiplicative in the number of scenarios (i.e., $|S||R|^2$). The number of variables

can easily be over one million when the number of repositories exceeds two hundred. Problems of this size can take up to 30 seconds to solve with a linear-programming solver and the resulting MIP can take several hours to complete. Our goal is thus to reduce the number of variables in the MIP solver significantly, without degrading the quality of the solutions too much.

The Aggregate Stochastic Storage Model (ASSM) is inspired by the structure of the solutions to the baseline algorithm. Customers are generally served by storage locations that are *nearby* and commodities are transported over large distances only in extreme circumstances. We exploit this observation by using a geographic clustering of the repositories. The clustering partitions the set of repositories R into C clusters and the repositories of a cluster $i \in C$ are denoted by CL_i. For a given clustering, we say that two repositories are *nearby* if they are in the same cluster; otherwise the repositories are *far away*. Nearby repositories have a tightly coupled supply and demand relationship and hence the model needs as much flexibility as possible in mapping the supplies to the demands. This flexibility is achieved by allowing commodities to flow between each pair of repositories within a cluster (as in SSM). When repositories are far away, the precise supply and demand relationship is not as crucial since the warehouse to customer relationship is calculated in the customer allocation stage of the algorithm. As a result, it is sufficient to reason about the aggregate flow moving between two clusters at this stage of the algorithm. The aggregate flows are modeled by introducing *meta-edges* between each pair of clusters. If some demand from cluster $a \in C$ must be met by storage locations from cluster $b \in C$, then the sending repositories CL_b pool their commodities in a single *meta-edge* that flows from b to a. The receiving repositories CL_a then divide up the pooled commodities in the *meta-edge* from b to meet all of their demands. Additionally, if each *meta-edge* is assigned a travel cost, the *meta-edge* can approximate the number of trips required between two clusters by simply dividing the total amount of commodities by the vehicle capacity, as is the case for all the other flow edges. Figure 2.8 indicates visually how to generate the flow decision variables for the clustered problem and how commodities can flow on *meta-edges* between customers in different clusters.

As stated above, the number of variables in the SSM is quadratic in the number of repositories. Given a clustering $CL_{i \in C}$, the number of variables in the clustered storage model is (1) quadratic within each cluster (i.e., $\sum_{i \in C} |CL_i|^2$); (2) quadratic in the number of clusters, (i.e., $|C|^2$); (3) and linear in the repositories' connections to the clusters (i.e., $2|R||C|$). The exact number of variables clearly depends on the clustering considered. However, given a specific number $|C|$ of clusters, a lower bound on the number of variables is obtained by dividing the repositories evenly among all the clusters, and the best possible variable reduction in a problem of size n with c clusters and s scenarios is $s\ (\frac{n^2}{c} + 2nc + c^2)$.

Given a clustering $CL_{i \in C}$ and cluster-to-cluster travel times CT_{scc} for each scenario, the ASSM is as presented in Model 5. The objective function has two terms for the delivery times, one for the shipping between repositories inside a cluster and one for shipping between clusters. Constraints (M5.1–M5.2) are the same as in the SSM model. Constraints (M5.3–M5.4) take into account the fact that the commodity can be shipped from repositories inside the clusters and from clusters.

Model 5 Aggregate Stochastic Storage (ASSM).

Let:

$CS_c = \sum_{i \in CL_c} RC_i$ – total storage in cluster c

$CQ_{sc} = \sum_{i \in CL_c} Q_{si}$ – total demand in cluster c in scenario s

Variables:

$Stored_i \in (0, RC_i)$ – quantity stored at repository i

$Open_i \in \{0, 1\}$ – non-zero storage at repository i

Second-stage variables for each scenario s:

$Unsatisfied_{si} \in (0, Q_{si})$ – unsatisfied demands at repository i

$Incoming_{sic} \in (0, Q_{si})$ – units shipped from cluster c to repository i

$Outgoing_{sic} \in (0, RC_i)$ – units shipped from repository i to cluster c

$Sent_{sij} \in (0, min(RC_i, Q_{sj}))$ – units shipped from repository i to repository j

$Link_{scd} \in (0, min(CS_c, CQ_{sd}))$ – units sent from cluster c to cluster d

Minimize:

$$\sum_{s \in S} P_s \sum_{i \in R} Unsatisfied_{si},$$

$$\sum_{s \in S} P_s (\sum_{c \in C} \sum_{i \in CL_c} \sum_{j \in CL_c} T_{sij}\ Sent_{sij}/VC + \sum_{c \in C} \sum_{d \in C} CT_{scd}\ Link_{scd}/VC),$$

$$\sum_{i \in R} (RI_i\ Open_i + RM_i\ Stored_i)$$

Subject To:

$$\sum_{i \in R} (RI_i\ Open_i + RM_i\ Stored_i) \leq B \tag{M5.1}$$

$$RC_i\ Open_i \geq Stored_i \qquad \forall i \in R \tag{M5.2}$$

$$\sum_{j \in R} Sent_{sji} + \sum_{c \in C} Incoming_{sic} + Unsatisfied_{si} = Q_{si} \qquad \forall s \in S\ \forall i \in R \tag{M5.3}$$

$$\sum_{j \in R} Sent_{sij} + \sum_{c \in C} Outgoing_{sic} \leq Stored_i \qquad \forall s \in S\ \forall i \in R \tag{M5.4}$$

$$\sum_{i \in CL_c} Outgoing_{sid} = Link_{scd} \qquad \forall s \in S\ \forall c \in C\ \forall d \in C \tag{M5.5}$$

$$\sum_{i \in CL_d} Incoming_{sic} = Link_{scd} \qquad \forall s \in S\ \forall c \in C\ \forall d \in C \tag{M5.6}$$

$$Sent_{sij} = 0 \qquad \forall s \in S\ \forall i \in DL_s\ \forall j \in R \tag{M5.7}$$

$$Outgoing_{sic} = 0 \qquad \forall s \in S\ \forall i \in DL_s\ \forall c \in C \tag{M5.8}$$

Constraints (M5.5–M5.6) aggregate the outgoing and incoming flow for a cluster, while constraints (M5.7–M5.8) express the damage constraints. Note that the array of variables $Sent_{sij}$ is sparse and includes only variables for repositories inside the same cluster (for simplicity, this is not reflected in the notation).

Objective Decomposition of Storage Allocation

The ASSM significantly decreases the number of variables, but it still requires creating a quadratic number of variables for each cluster. Since this is multiplied by the number of scenarios, the resulting number of variables can still be prohibitive for very large instances. This section presents an objective

Model 6 Phase 1 of the Lexicographic Stochastic Storage (LSSM-1).

Let:

$SQ_s = \sum_{i \in R} Q_{si}$ – total demand in scenario s

Variables:

$Stored_i \in (0, RC_i)$ – quantity stored at repository i
$Open_i \in \{0, 1\}$ – nonzero storage at repository i
$Used_s \in (0, SQ_s)$ – units used in scenario s

Minimize:

$$\sum_{s \in S} P_s \, (SQ_s - Used_s)$$

Subject To:

$$RC_i \; Open_i \geq Stored_i \qquad \forall i \in R \quad \text{(M6.1)}$$

$$\sum_{i \notin DL_s} Stored_i \geq Used_s \qquad \forall s \in S \quad \text{(M6.2)}$$

$$\sum_{i \in R} (RI_i \; Open_i + RM_i \; Stored_i) \leq B \qquad \text{(M6.3)}$$

decomposition that can be used when the objective is lexicographic, as assumed in this document and often the case in practice. Let us contemplate what this means for the behavior of the model algorithm as the budget parameter B is varied. With a lexicographic objective, the model first tries to meet as many demands as possible. If the demands can be met, it reduces delivery times until they can be reduced no further or the budget is exhausted. As a result, the optimization with a lexicographic objective exhibits three phases as B increases. In the first phase, the satisfied demands, routing times, and costs increase steadily. In the second phase, the satisfied demands remain at a maximum, the routing times decrease, and the costs increase. In the last phase, the satisfied demands remain at a maximum, the routing times remain at a minimum, and the costs plateau even when B increases further. The experimental results in Section 2.7 confirm this behavior.

The Lexicographic Stochastic Storage Model (LSSM) assumes that the objective is lexicographic and solves the first phase with a much simpler (and faster) model. The goal of this phase is to use the available budget in order to meet the demands as well as possible and it is solved with a two-stage stochastic allocation model that ignores the customer allocation and delivery time decisions. Since each scenario s has a total demand SD_s that must be met, it is sufficient to maximize the expected amount of demands that can be met, conditioned on the stochastic destruction of storage locations. Model 6 presents such a model.

During the first phase, LSSM-1 in Model 6 behaves similarly to the SSM for a lexicographic objective. But the model does not address the delivery times at all, since this would create a prohibitive number of variables. To compensate for this limitation, we use a second phase whose idea is summarized in the following greedy heuristic: if all the demands can be met, use the remaining budget to store as much additional commodity as possible to reduce delivery times. This greedy heuristic is encapsulated in another MIP model (LSSM-2) presented as Model 7. LSSM-2 utilizes the remaining budget while enforcing the decisions of the first step by setting the lower bound of the

Model 7 Phase 2 of the Lexicographic Stochastic Storage (LSSM-2).

Variables:

$StoredEx_i \in (Stored_i, RC_i)$ – quantity stored at repository i
$OpenEx_i \in \{0, 1\}$ – nonzero storage at repository i

Maximize:

$$\sum_{i \in R} StoredEx_i$$

Subject To:

$$RC_i \; OpenEx_i \geq StoredEx_i \qquad \forall i \in R \quad \text{(M7.1)}$$

$$\sum_{i \in R} (RI_i \; OpenEx_i + RM_i \; StoredEx_i) \leq B \qquad \text{(M7.2)}$$

$StoredEx_i$ variables to the value of the $Stored_i$ variables computed by LSSM-1. This approximation is rather crude but produces good results on actual instances (see Figures 2.21 and 2.22 in Section 2.7). Our future work will investigate how to improve this formulation by taking into account customer locations, while still ignoring travel distances.

The resulting approach is less flexible than the SSM and ASSM approaches because the time part of the multi-objective function is pushed into the second-stage variables and does not easily extend to a linear combination of the multi-objective function. However, it produces a significant improvement in performance by decreasing the number of decision variables from quadratic to linear. The asymptotic reduction is essential for scaling the algorithm to very large instances. Note that it is well known in the goal-programming community that lexicographic multi-objective programs can be solved by a series of single-objective problems [64]. The sub-objectives are considered in descending importance and, at each step, one sub-objective is optimized in isolation and side constraints are added to enforce the optimization in the previous steps. Our decomposed storage model follows the same schema, except that the second step is necessarily approximated due to its size.

2.4 Understanding the Decomposition Quality

In this section we discuss several possible shortcomings of the problem decomposition proposed in Section 2.2. Despite several attempts to improve the decomposition with sophisticated technology from the literature, we have not been able to improve solution quality significantly over the algorithm proposed in Section 2.2.

2.4.1 Alterative Warehouse Exploration

The selection of which warehouses to open is an important step for SCAPs because it directly determines the customer allocation problem and the quality of the tasks calculated in the repository routing stage. Once the repositories are selected, there is no opportunity for correction; any mistake in repository selection propagates into all stages of the algorithm.

This problem is well known in the location routing community because they often solve the

location routing problem by decomposition. The current state of the art in location routing problems (LRP) uses an iterative process in which the final solution is aggregated into a new problem and fed back into the warehouse location algorithm [83]. At a high level, the iterative algorithm does the following:

1. While (the set of open repositories has not reached a fixed point)
   (a) Assign the customers to repositories;
   (b) Solve the VRP for each repository;
   (c) For each tour t, make a supercustomer at the centroid of t with the sum of the demands;
   (d) Start over with the new set of supercustomers.

On standard LRP benchmarks this iterative repository exploration algorithm works well and often converges to an optimal set of repositories in two iterations. Unfortunately, we cannot apply this algorithm directly because SCAPs differ significantly from standard LRP benchmarks. First, the repositories and customers in SCAPs are not disjoint sets, so supercustomers cannot replace the original customers. Second, one customer may be split between several repositories. Third, each disaster scenario produces different trips, so each scenario has a unique set of supercustomers. Finally, the time matrix is only a metric space and not a Euclidean space, so it is not obvious how to determine the distance matrix for the supercustomers.

We tried various approaches to overcome these limitations. In particular, we associated supercustomers with each scenario and kept both original customers and supercustomers. Unfortunately, the experimental results were inconclusive and the iterative scheme could not find significantly better sets of repositories. Although the first-stage MIP uses a very coarse estimate of travel time, it seems to be sufficient for repository selection.

We also considered using the "squeaky wheel" optimization framework [65] to iteratively explore different warehouse allocation. At a high level, our squeaky-wheel approach is as follows:

1. For a fixed number of iterations
   (a) Solve the SCAP problem;
   (b) For each scenario s, select the vehicle v with the longest tour;
   (c) Increase the distance between all customer-warehouse pairs on v's tour by 10%;
   (d) Start over with the new distances.

This approach was not observed in any of the SCAP instances to decrease the longest tour length significantly.

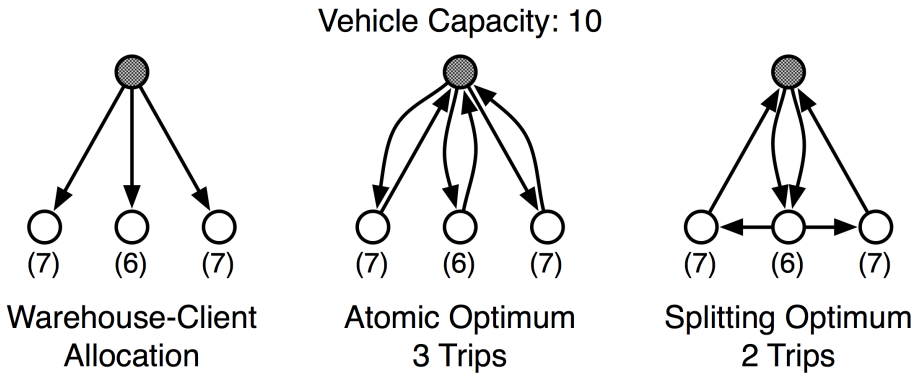

Figure 2.9: The Benefits of Splitting Deliveries.

2.4.2 Splitting Deliveries

So far, we have assumed that customer deliveries are atomic (i.e., all demands defined in the repository routing stage must be delivered at once). This is a common assumption in VRPs, but in some cases it yields sub-optimal tours, as demonstrated in Figure 2.9. We investigated experimentally if splitting deliveries would bring significant benefits to SCAPs. To determine the potential benefits, we considered a simple approach that does not scale well computationally but was sufficient for this purpose. In outline, the split-delivery algorithm is:

1. Enumerate all sets of customers;
2. For each set, use dynamic programming to find an optimal routing time of those customers;
3. Use trip-based tree search to find an optimal set of trips with demand splitting.

It is clear that this approach does not scale, since enumerating all the sets of customers takes 2^c time, where c is the number of customers. Fortunately, many SCAP benchmarks have relatively small numbers of customers per repository, making this model viable. Despite the effort to reduce the number of delivery tasks, the experimental results indicate that this approach does not bring significant benefits on current SCAP benchmarks. We hypothesize that this is due to the large number of repositories in SCAP and the splitting that occurred in the first stage of the algorithm.

2.4.3 Benefits of Customer Allocation

It may be hard to believe that a small change in the objective function in the customer allocation stage brings significant benefits over the linear relaxation calculated by the stochastic storage problem. However, our benchmarks have demonstrated that this simple change can improve the last delivery time by as much as 16%. This is the most significant improvement observed in all the enhancements to the SCAP algorithm.

2.5 Solving Real-World Routing Problems

The MIP models presented in Section 2.2 can quickly solve problems with about 15 tasks. However, once the number of tasks exceeds 20, the problems become nearly intractable. Even worse, real-word problems often have 100 tasks or more, well beyond the tractable range. These difficulties were also faced in [23]. Our solution to these computational challenges is threefold. First, we solve the repository routing problem using a configuration-based model that is exponential in the worst case but very effective in practice. Second, we approximate the fleet routing problem with Large Neighborhood Search (LNS) [92], which is well known to be effective for VRP problems. Last, we boost the performance of LNS by using an aggregate fleet-routing step to find a high-quality initial solution.

2.5.1 Set-Based Repository Routing

The extraction of full trips in the repository routing decomposition produces VRPs with properties that are uncommon in traditional VRPs. Specifically, customer demands that remain after removing full-capacity trips are distributed roughly uniformly through the range $0..VC$. This property allows a repository-routing formulation that scales much better than the pure MIP formulation described earlier. Indeed, if the customer demands $d_1, \ldots, d_c$, are uniformly distributed in the range $0..VC$, the expected number of sets satisfying the vehicle capacity is smaller than c^3 when c is not too large (e.g., $c \leq 50$). This observation suggests using a set-covering approach similar to that in [97]. The formulation is as follows:

1. Enumerate all customer sets satisfying the capacity constraint;
2. Use dynamic programming to calculate the optimal trip for those customer sets;
3. Use MIP to find a partition of customers with minimal delivery time.

This hybrid model is more complex, but each subproblem is small and it scales much better than the pure MIP model.

2.5.2 Aggregate Fleet Routing

For instances with less than 200 repositories, the most computationally intense phase of the algorithm is the fleet routing. We now investigate how to initialize the LNS search with a high-quality solution. Recall that the fleet routing problem associates a node with every trip. Given a scenario s, a lower bound for the number of trips is

$$\sum_{i \in R} \sum_{j \in R, i \neq j} \frac{Sent_{sij}}{VC}$$

Clearly, the size and complexity of this problem grow with the amount of commodities moved and are very sensitive to the vehicle capacity VC. To find high-quality solutions to the fleet routing subtask, we aggregate the trips to remove the dependence on the amount of commodities delivered. More

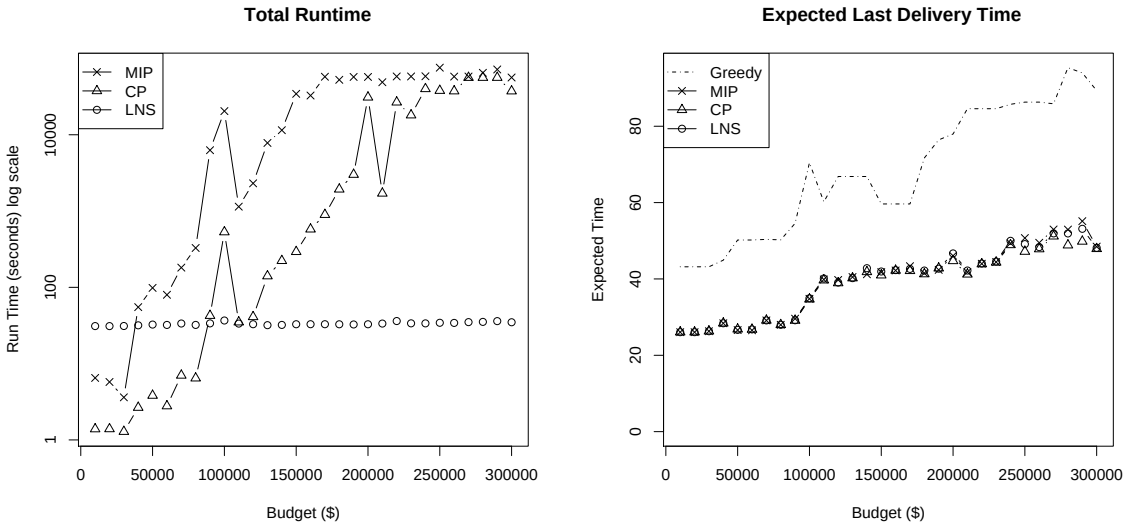

Figure 2.10: Quality and Run Time Comparison of LNS, CP, and MIP for the Fleet Routing Problem.

precisely, we define an aggregate fleet routing model in which all trips at a repository are replaced by an aggregate trip whose service time is the sum of the trip service times at that repository. The number of nodes in the aggregate problem is now proportional to the number of repositories. Finding a good initial solution is not important for smaller problems (e.g., $|R| \approx 25, |V| \approx 4$) but becomes critical for larger instances (e.g., $|R| \geq 100, |V| \geq 10$). Since the aggregate problem is much simpler, it obtains high-quality solutions quickly.

2.5.3 Large-Neighborhood Search for Fleet Routing

The fleet routing problem can be solved using large neighborhood search (LNS) to obtain high-quality solutions quickly for the significant number of nodes arising in large instances. LNS, a hybrid of constraint programming (CP) and local search, can be summarized as follows.

1. Find a feasible solution using constraint programming;
2. Relax a portion of the best-known solution and search for an improving, feasible solution by reassigning the relaxed variables;
3. Iterate step 2 until no improvement is found or some termination criterion is met.

LNS combines the strength of constraint programming in exploring small combinatorial spaces effectively with the benefits of local search to find high-quality solutions quickly. For the fleet routing application, at each optimization step, the LNS algorithm selects 15% of the trips to relax, keeping the rest of the routing fixed. The neighborhood exploration has a termination criterion of $(0.15|Nodes_s|)^2$ backtrackings. The relaxation and re-optimization step is repeated until an instance-specific time limit is reached.

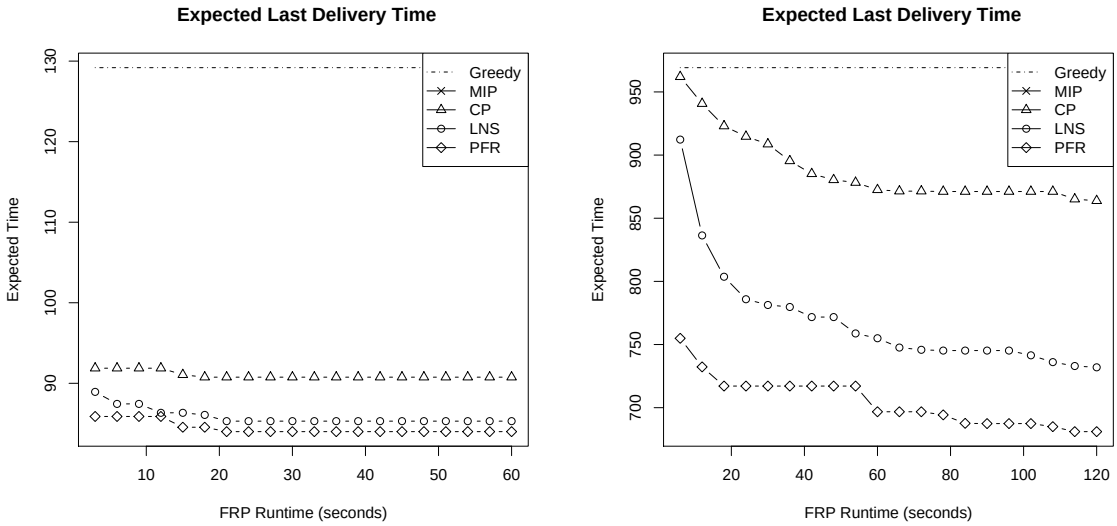

Figure 2.11: Solution Quality Change Over Time: Comparison of PFR, LNS, CP, and MIP for the Fleet Routing Problem.

Figure 2.10 indicates the time and quality tradeoffs among LNS, CP, and MIP for various budget parameters on our smallest benchmark, BM1. Recall from Section 2.5.2 that routing problem difficulty increases with the budget, and thus it is natural that the algorithm run times do the same. The left graph shows the run time of the fleet routing algorithm on a logarithmic scale. The LNS algorithm has a fixed timeout of 60 seconds, while the MIP and CP approaches are limited to 24 hours. It is obvious that the CP algorithm is generally an order of magnitude faster than the MIP. However, both approaches are too slow for budgets exceeding $200,000. The truly surprising fact is presented on the graph of fleet routing solution quality. Despite the very short running time, LNS finds an optimal or near-optimal solution for all budget settings. This is very encouraging because the instances where MIP and CP are viable are few compared to those that the U.S. government would like to consider.

It is clear that LNS can find high-quality solutions quickly, but Figure 2.10 does not indicate how much of the time used by the MIP and CP formulations is necessary for the proof of optimality. It may be possible that MIP and CP find the optimal solution quickly and spend the bulk of the time on the proof. Figure 2.11 investigates how the solution quality of each algorithm changes over time. In all cases, we only consider run times up to the specified LNS time limit. The figure compares five possible approaches to the fleet routing problem (FRP): (1) the greedy approach discussed in detail in Section 2.6.4, (2) the MIP formulation, (3) the CP formulation, (4) the LNS formulation, and (5) the PFR formulation, which is the LNS formulation extended with aggregate fleet routing. In all the experiments, the MIP formulation was unable to find a feasible solution within the time limit and does not appear in Figure 2.11. The left graph shows the algorithms on the smallest benchmark, BM1, with a budget of $600,000; run times range from 3 to 60 seconds. The results indicate that

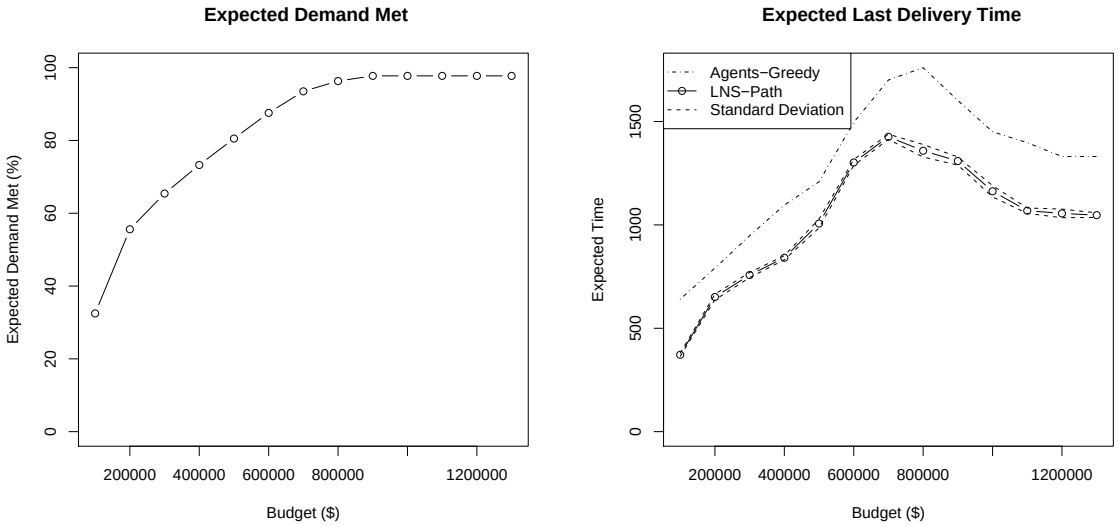

Figure 2.12: The Impact of Varying the Budget.

the CP formulation quickly gets stuck in a low-quality solution and much more time is required for a high-quality solution. BM1 is small, so the PFR algorithm is only slightly better than the simpler LNS approach. The right graph shows the algorithms on a medium-sized benchmark, BM4, with a budget of $300,000; run times range from 6 to 120 seconds. Again, the results indicate that the CP formulation quickly gets stuck in a low-quality solution and much more time is required to find a high-quality solution. This graph also indicates the significant benefits of LNS and PFR on larger benchmarks. For even larger benchmarks, such as BM5, the CP algorithm begins to have difficulty finding a feasible solution within the LNS time limit (just as the MIP did with smaller benchmarks). The LNS algorithm with aggregate fleet routing is essential for solving the largest benchmarks the U.S. government would like to consider, and is used for all the results reported in Section 2.7.

2.6 Practical Applications

So far we have focused on the algorithmic details of solving SCAP problems. Although these details are interesting from an academic perspective, the true motivation of this work is to aid policy makers in making decisions about disaster management. In this section we discuss several key applications of this algorithm that can provide decision support to policy makers and were directly requested by policy makers of the U.S. government.

2.6.1 Strategic Budget Decisions

Every year policy makers must decide how much money should be used for disaster preparedness and recovery operations. Without rigorous scientific study, the potential consequences and risks are

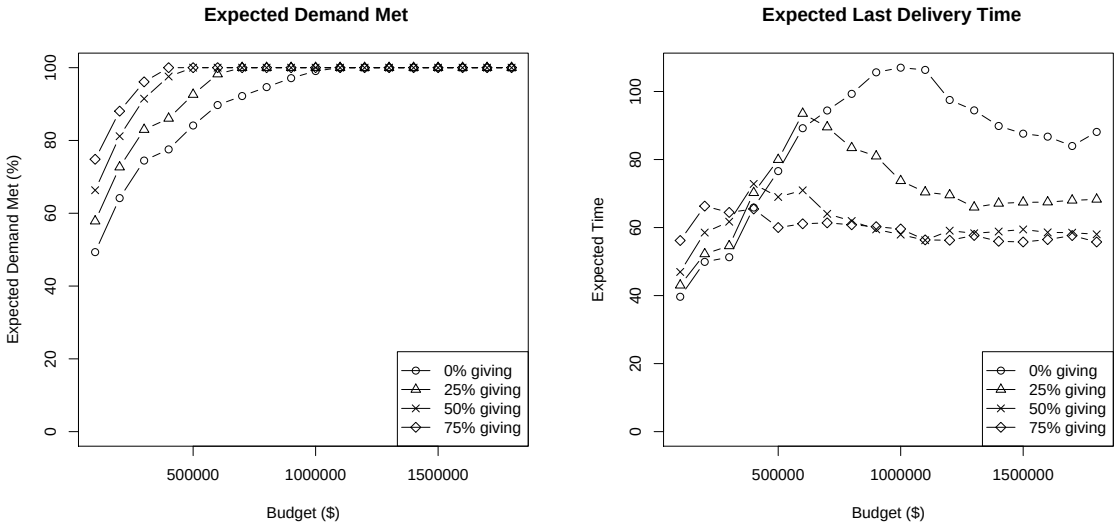

Figure 2.13: The Benefits of Corporate Giving with a Central Location.

unclear and as a result policy makers must resort to the philosophy,"the more money allocated to preparedness the better". However, budgets are limited and decisions on preparedness allocation become increasingly difficult when funds must be taken from other public services such as infrastructure maintenance and education programs. Fortunately, this SCAP algorithm can aid policy makers in making budgetary dissections by quantifying the quality of recovery operations under different budgets. Figure 2.12 demonstrates how to provide budgetary decision support to policy makers. In both graphs, the budget is varied on the x-axis and performance metrics are quantified on the y-axis. The left graph shows the percentage of demands that can be met. A policy maker can clearly see that at least a budget of about \$900,000 is required if all demands are to be satisfied. The right graph shows the time of the last delivery. Once all demands can be satisfied, this graph indicates how additional budget will reduce the time of the last delivery. A policy maker can see that the delivery time can be reduced significantly up to a budget of \$1,100,000, but additional funds bring few benefits. The SCAP algorithm for these kinds of decision support studies can be computed quickly and shows policy makers the quantifiable consequences of various budgetary decisions.

2.6.2 Effects of Corporate Giving

After a catastrophic disaster, corporations often make donations to aid federal organizations in relief efforts, so that a study of corporate giving is of particular interest to policy makers. Fortunately, the SCAP model can easily support analysis of corporate giving simply by modifying the problem input. Once these modifications are made, the SCAP algorithm can be used to understand how corporate giving affects disaster planning and response.

Here we investigate how different quantities and locations of donations affect disaster response.

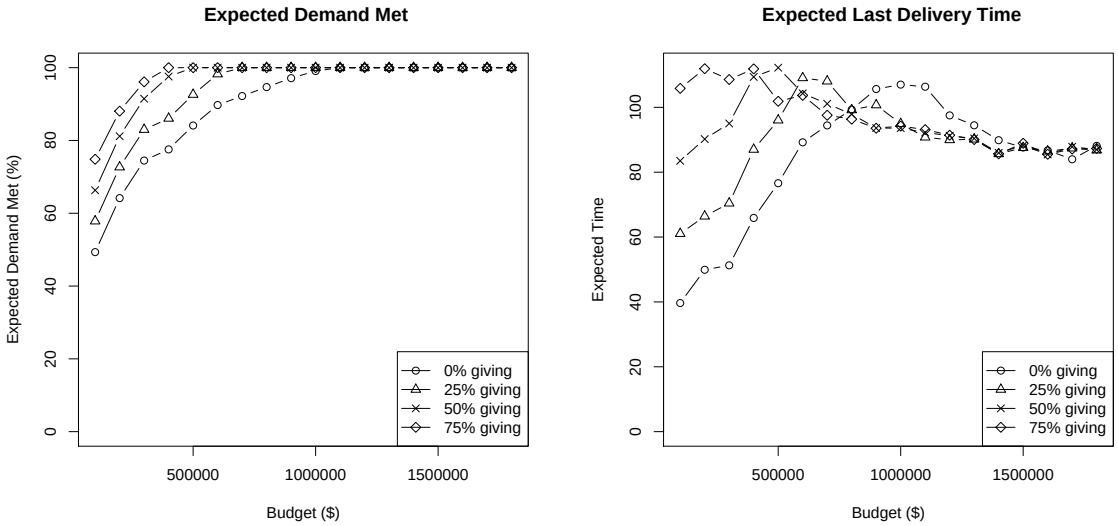

Figure 2.14: The Benefits of Corporate Giving with a Distant Location.

We consider donations of 25%, 50%, and 75% of the total demand, and compare them with no donations (i.e., 0% of the demand). Figure 2.13 presents the behavior when the donation site is *near* the center of demand (e.g., a large retail store location). The left graph shows the demand met as the budget increases and the effect of corporate giving is what you would expect: it enables more demands to be met when the budget is too small. The right graph depicts the last delivery time as the budget increases. These results are quite interesting. Initially, for low budgets, the maximum delivery time increases as more of the demand is met; as the budget increases, the last delivery time decreases substantially because the giving location is placed close to the demand locations.

Figure 2.14 presents the behavior when the donation site is *far away* from the demands (e.g., a regional supply warehouse). The results are identical for the demand (left graph). However, the results for the latest delivery times are fundamentally different. Until the demand is met, the latest delivery times increase, as is normal. However, the increase is significantly larger than for a central location. When the budget is large, there is no benefit to corporate giving, even though it had significant benefits from a central location. It is only for budgets that meet all the demand but are not very large that corporate giving reduces the latest delivery time. This analysis quantifies the benefits of corporate giving in monetary terms and may help decision makers in influencing corporate giving.

2.6.3 Robust Optimization

So far the SCAP algorithm has considered minimization of the expected outcome of the stochastic set of disasters. Specifically, the Stochastic Storage Model considers the objective function

$$\text{Minimize: } \sum_{s \in S} P_s \text{UnsatisfiedDemands}_s, \sum_{s \in S} P_s \text{TravelTime}_s, \text{StorageCosts}$$

This is a natural formulation but policy makers may be interested in minimizing the worst-case outcome instead of the average-case outcome, as with the delivery-time objective in the routing aspect of the SCAP problem. This alternate objective is known as robust optimization. The objective above can be formulated as a robust optimization problem as:

$$\text{Minimize: } \max_{s \in S} \text{UnsatisfiedDemands}_s, \max_{s \in S} \text{TravelTime}_s, \text{StorageCosts}$$

The SSM is easily adapted for this objective in two steps. First we introduce auxiliary variables M^d,M^t and add the constraints

$$M^d \geq \text{UnsatisfiedDemands}_s \ \ \forall s$$

$$M^t \geq \text{TravelTime}_s \ \ \forall s$$

then the objective becomes

$$\text{Minimize: } M^d, M^t, \text{StorageCosts}$$

Together these modifications allow the SSM to solve a robust optimization variant of the SCAP problem. Both objective formulations are worth consideration and we allow the policy makers to choose the kind of objective formulation they prefer.

2.6.4 Recovery Practice in the Field

Due to the complexity of disaster recovery problems, it is often impossible to calculate a globally optimal solution. However, there is often some measure of uncertainty in the problem specification so a global optimal solution is not crucial in this case. In this context we shift our goal from finding an optimal solution to simply improving the best current field practice. In the rest of this section we discuss our approximation of the best field practices for the SCAP problem.

The SCAP problem can be viewed simply as storage decisions (first-stage variables) and routing decisions (second-stage variables). In developing an algorithm of best field practices, we replace each of these decisions with an algorithm similar to the ones currently used by policy makers. First we consider the storage decisions. Despite considerable effort, we have found no clear documentation on how policy makers make storage decisions for disaster preparation. In lieu of this information, we approximate the storage decisions by giving our best-practice algorithm the same storage decisions as the SCAP algorithm. This is highly optimistic in that, since that stage is solved to optimality, policy makers could not make better allocation decisions. Second, we consider the routing decisions. In the context of disaster recovery, truck drivers are assigned to specific trucks and possibly specific

areas and work more or less independently to deliver as much commodity as possible. We can simulate this behavior using an agent-based algorithm where each driver is an agent in the system and greedily makes pickups and deliveries as fast as possible. The agent decision process is as follows:

```
GREEDY-TRUCK-AGENT (GTA)()
1  while there exists some commodity to be picked up and demands to be met
2     do if I have some commodity
3           then drop it off at the nearest demand location
4           else  pick up some commodity from the nearest warehouse
5  goto final destination
```

While this agent-based algorithm roughly approximates current relief delivery procedures, it is in fact optimistic because it assumes all the drivers have global and up-to-date information about the state of the stored commodities and demand locations. This is not an unreasonable as the recovery efforts can be orchestrated from a central office, but in practice perfect coordination of their efforts is unlikely. Together these algorithms permit a solution to the SCAP problem that is an optimistic approximation of current practices in the field. We call this approximation the GTA algorithm and it will serve as the basis of comparison for the evaluation of the SCAP algorithm in Section 2.7.

2.7 Benchmarks and Results

2.7.1 Benchmarks

The benchmarks were produced by Los Alamos National Laboratory and are based on the U.S. infrastructure. The disaster scenarios were generated by state-of-the-art hurricane simulation tools similar to those used by the U.S. National Hurricane Center. The benchmark sizes and algorithm parameters are presented in Table 2.1. The *Trip Lower Bounds* are simply the total amount of commodities that are shipped divided by the vehicle capacity (as discussed in Section 2.5.2). These values are included because they are a good metric for the routing difficulty of a benchmark. However, the amount of commodities that need to be moved can vary significantly from scenario to scenario, and we thus present both the smallest and the largest trip bounds across all the scenarios. Benchmarks 3 and 6 feature scenarios in which the hurricane misses the region; this results in the minimum trip bound being zero. This is important, since any algorithm must be robust with respect to empty disaster scenarios which arise in practice when hurricanes turn away from shore or weaken prior to landfall. All the experimental results vary the value of the budget B to evaluate the algorithm under different conditions. The results are consistent across multiple weight configurations, although there are variations in the problem difficulties. It is also important to emphasize that, on these benchmarks, the most difficult routing problems have a size greater than the *Max Trip Lower Bound* and thus yield considerably large VRPs.

Benchmark	n	m	a	Min Trip Lower Bound	Max Trip Lower Bound	CA Timeout	LNS Timeout	Clusters
BM1	25	4	3	6	27	25	10	4
BM2	25	5	3	60	84	25	20	4
BM3	25	5	3	0	109	25	20	4
BM4	30	5	3	35	109	30	20	4
BM5	100	20	3	82	223	100	200	4
BM6	25	5	18	0	140	25	20	4
BM7	30	10	18	7	23	30	20	4
BM9	250	10	18	7	23	250	90	10
BM10	500	20	18	13	45	-	180	-
BM12	1000	20	3	64	167	-	300	-

Table 2.1: SCAP Benchmark Statistics.

2.7.2 The Algorithm Implementation and the Baseline Algorithm

The final algorithm was implemented in the COMET system [43] and the experiments were run on Intel Xeon CPU 2.80 GHz machines running 64-bit Linux Debian. To validate our results, we compare our delivery schedules with an approximation of the recovery practice in the field as described in Section 2.6.4.

2.7.3 Evaluation of the Approach

We now evaluate the efficiency and quality of the approach.

Efficiency Results

Table 2.2 depicts the runtime results for BM1–BM9 (BM10 and BM12 are discussed later). In particular, the table reports, in averages, the total time in seconds for all scenarios (T_1), the total time when the scenarios are run in parallel (T_∞), and the times for the storage model (STO), the client-allocation model (CA), the repository routing (RR), the aggregate fleet routing (AFR), and fleet routing (FR). The first three fields (T_1, T_∞, STO) are averaged over ten identical runs on each budget parameter. The last four fields (CA, RR, AFR, FR) are averaged over ten identical runs on each budget parameter and each scenario. Since these are averages, the times of the individual components do not sum to the total time. The results show that the approach scales well with the size of the problems and is a practical strategy for solving SCAPs.

Quality of the Results

Table 2.3 depicts the improvement of our SCAP algorithm over the baseline algorithm. Observe the significant and uniform benefits of our approach, which systematically delivers about a 50% reduction in delivery time. Table 2.4 describes the correlations between the distances in the customer allocation and fleet routing models. The results show strong correlations, indicating that the distances in the customer allocation model are a good approximation of the actual distances in the fleet routing

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	89.89	21.90	39.96	13.01	0.9293	0.4670	9.257	0.1746	10.057	10.00
BM2	169.1	35.93	66.02	10.47	0.5931	0.2832	16.67	0.1956	19.26	20.00
BM3	98.58	14.51	61.07	13.79	0.3557	0.1748	7.225	0.1050	12.04	13.33
BM4	184.2	26.25	68.76	5.163	0.8892	0.3940	21.24	0.2075	19.58	20.00
BM5	1308	62.01	520.5	32.70	46.70	21.31	100.0	1.225	128.0	200.0
BM6	723.5	58.76	75.34	3.079	5.165	3.076	10.81	0.1281	13.35	15.56
BM7	832.0	97.05	75.13	13.31	16.15	5.153	5.500	0.4509	19.31	20.00
BM9	16091	13637	11078	13435	10643	13434	143.6	1.376	65.96	90.00

Table 2.2: SCAP Benchmark Runtime Statistics (seconds).

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9
Improvement(%)	56.6	43.0	73.8	54.1	60.7	90.8	55.6	261

Table 2.3: Improvements over the Baseline Algorithm.

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9
Correlation	0.9340	0.9995	0.9969	0.9981	0.9080	0.9888	0.9449	0.9927

Table 2.4: Correlations for the Distances in Customer Allocation and Fleet Routing.

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9
Absolute Difference	9.49	44.8	25.69	45.4	633	63.3	396	144
Relative Difference(%)	13.6	9.52	5.40	7.55	42.1	20.4	49.3	31.7

Table 2.5: The Difference in Delivery Times Between Vehicles.

model. Table 2.5 reports the absolute and relative differences between vehicle tour lengths in the solutions. They indicate that the load is nicely balanced between the vehicles when the number of vehicles is small (i.e. $m \approx 5$). More precisely, the maximum delivery times are often within 15% of each other on average for Benchmarks 1 through 4, strong evidence of the quality of our solutions. Benchmarks 5 through 9 have an abundance of vehicles (i.e. $m \geq 10$). These benchmarks model emergency response at a state level, not at a city level as in Benchmarks 1 through 4. In state-scale models, some vehicles have significantly reduced load because they would have to travel to the other side of the state to acquire more load, thus increaseing the maximum delivery time.

Behavioral Analysis

Figure 2.15 presents a behavioral analysis on Benchmark 5 (other benchmarks are consistent and are omitted for space reasons). The graph on the left shows how the satisfied demand increases with the budget, while the graph on the right shows how the last delivery time changes. Given the weight selection, it is expected that the demand and routing time will increase steadily with the budget until the total demand is met. At that point, the demand should stay constant and the routing time should decrease. The results confirm these expectations, and also indicate the significant benefits of our approach over the greedy routing algorithm. The benefits also increase as the budget increases, but are very significant even for low budgets.

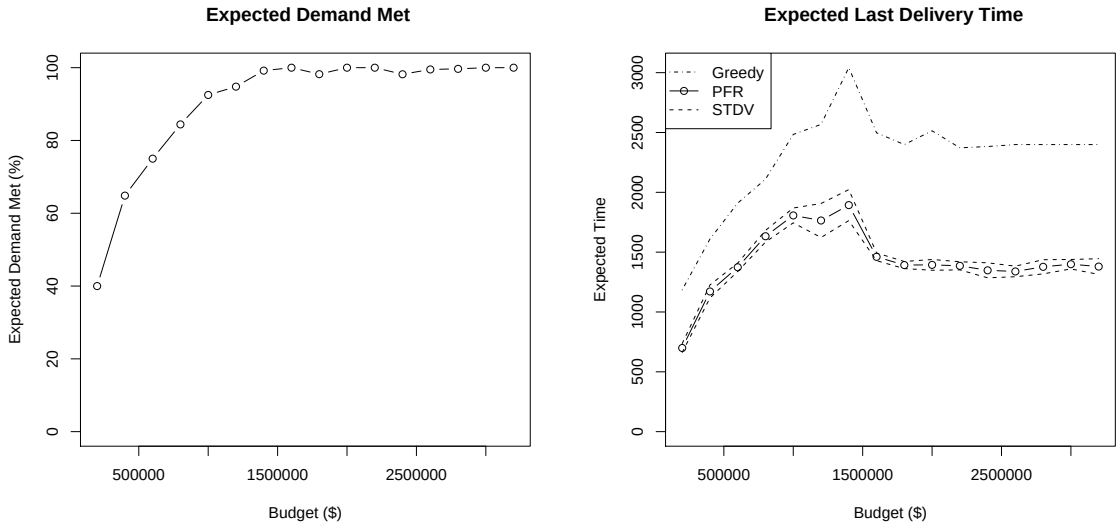

Figure 2.15: The Impact of Varying the Budget on Benchmark 5.

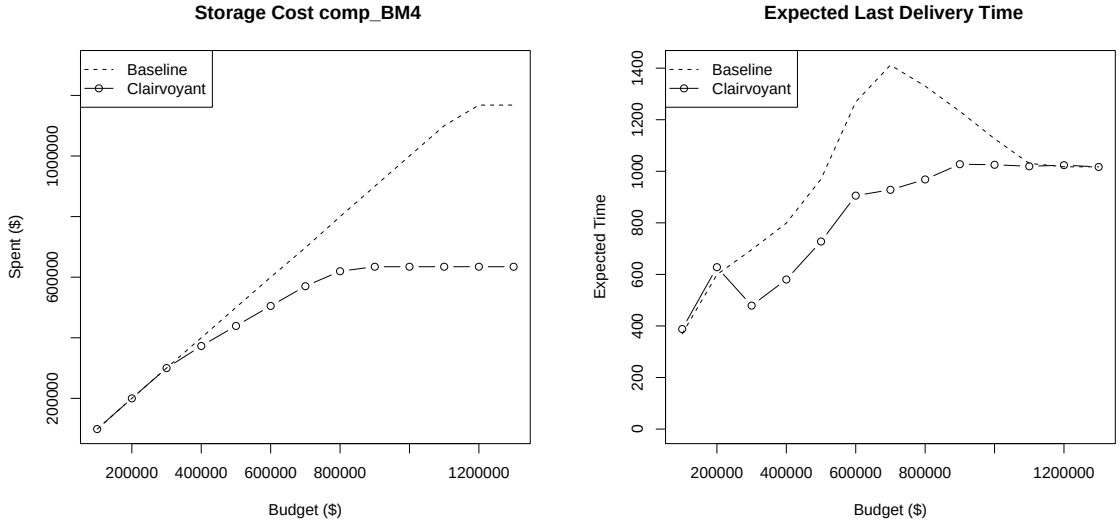

Figure 2.16: The Value of Perfect Information.

The Value of Perfect Information

Figure 2.16 presents the value of perfect information (i.e., the gain from knowing which disaster will occur) on Benchmark 4. The left graph shows storage cost as a function budget for the stochastic storage model and the clairvoyant model that has knowledge of the scenario. For smaller or even medium budgets, the clairvoyant does not bring substantial benefits. However, as the budget increases, the difference between the stochastic and deterministic version becomes significant.

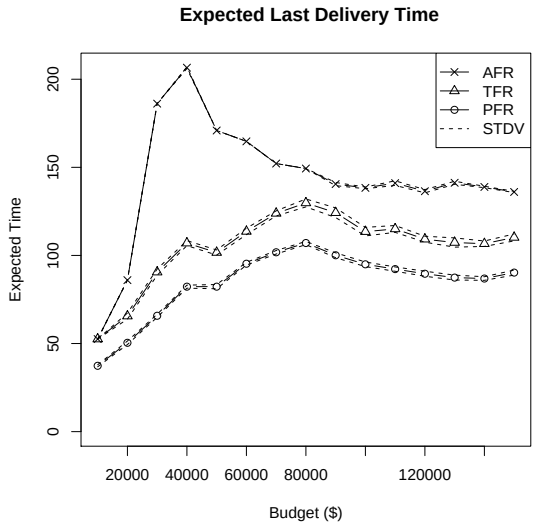

Figure 2.17: Comparing Various Algorithms for Fleet Routing.

The right picture depicts the same information for the last delivery time. Here, as the budget becomes very large, the stochastic and deterministic variants converge on the same value, since there are enough funds to fill all the storage facilities to capacity, and the solution is robust across all scenarios. The difference in the latest delivery peaks for mid-size budgets.

2.7.4 Study of the Algorithmic Improvements and Variations

Fleet Routing

Figure 2.17 presents experimental results comparing aggregate (AFR), tour-based (TFR), and path-based (PFR) fleet routing. (Only Benchmark 1 is presented; other results are consistent.) The figure depicts the significant improvements obtained by considering paths instead of tours; the improvements are consistent over all budgets. Equally interesting, by comparing AFR to TFR, the figure indicates the benefits of allowing different vehicles to perform the trips of a single repository. This flexibility brings significant improvements in quality for all budgets. Moreover, for budgets that allow only a few warehouses to be opened, the improvements to the latest delivery times are extremely significant. In these circumstances, AFR suffers significantly because warehouses typically carry uneven task sizes due to their capacity constraints.

Customer Allocation

Figure 2.18 presents the benefits of separating customer allocation from storage decisions. As mentioned earlier, the benefits are negligible when the budget is small. However, they become significant when the budget increases and can produce a reduction of up to 16% in the expected maximum delivery time.

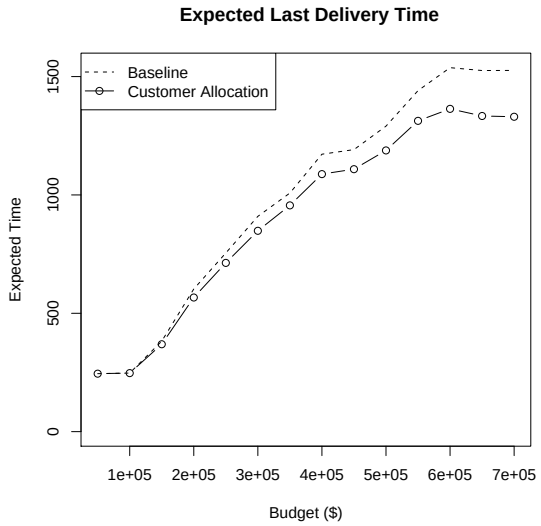

Figure 2.18: The Benefits of Decoupling Storage and Customer Allocation.

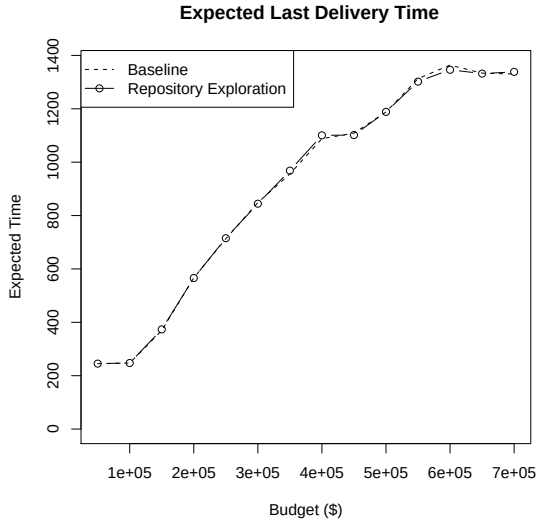

Figure 2.19: The Benefits of Iterative Repository Exploration.

Iterative Exploration

Figure 2.19 presents the experimental results on Benchmark 2 of the iterative repository exploration algorithm described in Section 2.4.1. As mentioned earlier, the benefits are marginal at best, and this variation is omitted in the final version of the algorithm.

Figure 2.20: The Benefits of Customer Splitting.

Customer Splitting

Figure 2.20 presents the experimental results of the customer-splitting algorithm described in Section 2.4.2 on Benchmark 1. The results of both algorithms are nearly identical and when they do differ the improvement is not significant. This slight improvement in quality is not justified by the high computational cost of the customer-splitting formulation. It is important to note that the splitting results may be biased by the lexicographic objective function. However, we have not found an objective variation where splitting brings significant benefits. Unfortunately, a comprehensive study is outside the scope of this work.

2.7.5 Very Large-Scale Instances

We now consider very large-scale instances (Benchmarks BM10 and BM12).

ASSM Quality and Efficiency Results

Table 2.6 depicts the improvement of our ASSM for the SCAP algorithm. Observe that ASSM (i.e. *STO*), runs twice as fast on Benchmarks 1 through 7 and 14 times faster on Benchmark 9. In this study repository clusters were generated using geographic locations (i.e. a Euclidean metric) and the best solution from ten samples of a traditional K-means algorithm. The sample with the smallest mean sum is used in the clustered storage model. The distances between clusters are calculated on a scenario-by-scenario basis using the average distance between all pairs of points in each cluster.

The runtime benefits of the clustering algorithm are largely due to the reduction in the number of variables in the model. Section 2.3.2 analyzed the variable reduction and pointed out that the reduction is tightly coupled with the clustering. Due to geographic considerations in these instances,

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	89.77	22.19	39.25	13.28	**0.5464**	**0.2389**	9.043	0.1791	10.04	10.12
BM2	169.4	35.91	65.93	10.49	**0.4846**	**0.1850**	16.81	0.2084	19.26	20.00
BM3	98.73	14.49	61.15	13.81	**0.3986**	**0.1609**	7.245	0.1092	12.05	13.33
BM4	182.8	24.28	69.74	3.822	**0.6950**	**0.3717**	20.88	0.2122	19.56	20.00
BM5	1266	70.41	487.4	35.18	**18.40**	**7.700**	100.0	0.8691	123.9	200.0
BM6	714.86	59.04	73.28	1.032	**3.130**	**1.041**	10.57	0.09642	13.27	15.56
BM7	823.6	98.79	67.95	12.99	**8.849**	**2.666**	5.479	0.4475	19.28	20.00
BM9	6377	803.6	1184	363.8	**747.7**	**363.5**	153.9	0.8491	66.72	90.00
BM10	-	-	-	-	-	-	-	-	-	-
BM12	-	-	-	-	-	-	-	-	-	-

Table 2.6: Clustered Storage Runtime Statistics (Seconds).

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
SSM	1875	1875	1875	2700	30000	11250	16200	1125000	-	-
ASSM	1116	1206	1248	1470	12246	7344	9576	237420	-	-
Lower Bound	1101	1101	1101	1443	9948	6606	8658	204300	-	-

Table 2.7: Clustered Problem Size Compared to the Baseline Algorithm.

the clustering exhibits great variation from instance to instance and it is important to report the actual reduction in problem size. Table 2.7 presents the number of variables of the SSM and the ASSM, as well as the lower bound on the number of variables. Observe that the benefits become more significant as the problem size grows; the runtime results confirm this.

Table 2.8 describes the relative changes in routing times from the SSM. The quality degradation of the greedy algorithm is also presented to provide a sense of scale. Because the ASSM is a coarser approximation of the travel time, some decrease in routing quality is expected. It is impressive that the reduction in quality is not significant (especially when compared with the greedy algorithm).

It is surprising that sometimes the clustering model improves the quality of the routing solution, because the travel time objective is only approximated in *all* of the stochastic storage models. When there are large distances between nodes, the ASSM's meta-edges provide a more accurate estimate of the number of trips needed between two clusters. Unfortunately, this model still suffers from the same memory issues as the SSM and is unable to solve Benchmarks 10 and 12.

SSSM Quality and Efficiency Results

Table 2.9 depicts the improvement of our SSSM for the SCAP algorithm. Observe the consistent reduction in run time of the storage model (STO): on Benchmark 9 runs about 1000 times faster than the SSM.

Table 2.10 describes the relative change in routing times from the SSM. The quality degradation of the greedy algorithm is also presented to provide a sense of scale. Because the SSSM has no information about the travel time, some decrease in routing quality is expected. Again, it is impressive that the reduction is so small (especially when compared with the greedy algorithm). This model is the first that can solve Benchmarks 10 and 12, and thus can report the relative improvement over

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
Relative Change(%)	-0.356	0.0834	-0.108	-0.504	-0.887	3.54	-0.308	2.95	-	-
Greedy Change(%)	56.4	43.1	73.7	52.0	64.0	92.2	55.5	259	-	-

Table 2.8: Clustered Quality Compared to the Baseline Algorithm

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	94.97	24.38	41.32	12.79	**0.1166**	**0.06717**	11.46	0.1943	9.819	10.00
BM2	169.3	36.04	65.59	10.43	**0.1624**	**0.07953**	16.86	0.2375	19.25	20.00
BM3	98.85	14.31	60.97	13.76	**0.1705**	**0.09249**	7.280	0.1260	12.12	13.34
BM4	183.8	24.26	69.15	3.527	**0.2514**	**0.1918**	21.27	0.2547	19.59	20.00
BM5	1240	69.07	468.6	33.70	**2.125**	**0.8614**	100.0	0.8141	120.6	200.0
BM6	719.8	56.91	70.46	0.08338	**0.3516**	**0.08734**	11.06	0.08603	13.23	15.56
BM7	810.5	100.8	59.54	13.51	**0.7089**	**0.1497**	5.397	0.3703	19.17	20.00
BM9	5859	458.6	488.9	26.32	**53.84**	**27.30**	164.8	0.7035	65.92	90.00
BM10	32048	1708	1921	108.9	**17.34**	**22.02**	1385*	8.659	175.7	180.0
BM12	18201	73.11	6146	46.54	**14.12**	**0.2223**	5485*	14.28	227.3	300.0

Table 2.9: Decomposed Storage Runtime Statistics (Seconds).

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
Relative Change(%)	3.47	-0.891	0.165	0.919	3.38	7.02	-0.0884	6.77	-	-
Greedy Change(%)	61.1	42.5	74.5	53.5	67.0	103	55.5	295	78.5	91.5

Table 2.10: Decomposed Quality Compared to the Baseline Algorithm.

the greedy algorithm. However, it cannot report the relative change compared to the SSM because that model cannot solve these instances.

Some policy makers may be concerned by the 7.0% increase in delivery time in benchmark 6 and may prefer to use the SSM. However, some types of disasters require immediate response and every minute is valuable. In those extreme situations, the decomposed storage model is a much faster alternative to the SSM. These new algorithms allow the policy maker to choose on a case-by-case basis which is preferable: a faster response or a higher-quality solution.

The lack of information about travel time is an advantage for the memory usage of the SSSM. Only three pieces of the problem specification need to be considered, the repository information, scenario demands, and scenario damage. This resolves the memory issues faced by the other models by loading the scenario travel time separately for each scenario. This allows the SSSM to scale to the largest benchmarks. Figure 2.21 visually summarizes the run time and quality tradeoffs among the SSSM, ASSM, and SSM.

Due to the enormous size of Benchmarks 10 and 12, the customer allocation stage of the algorithm does not return a feasible solution within 1000 seconds. To resolve this difficulty, we simply ignore the integer variables and solve the linear programming relaxation of the same problem, which is then rounded. As Table 2.9 indicates, just solving a linear programming relaxation of these problems can take over 10 minutes. Additionally, to make the run time of the SSSM stable on the largest instances, the solver is terminated whenever the optimality gap is reduced to 0.05%.

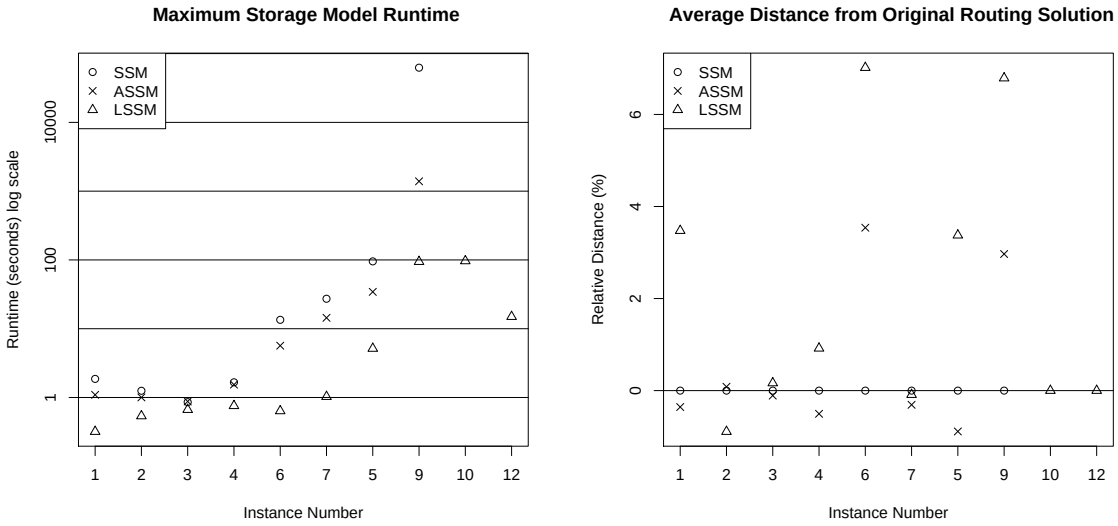

Figure 2.21: Runtime and Quality Tradeoffs.

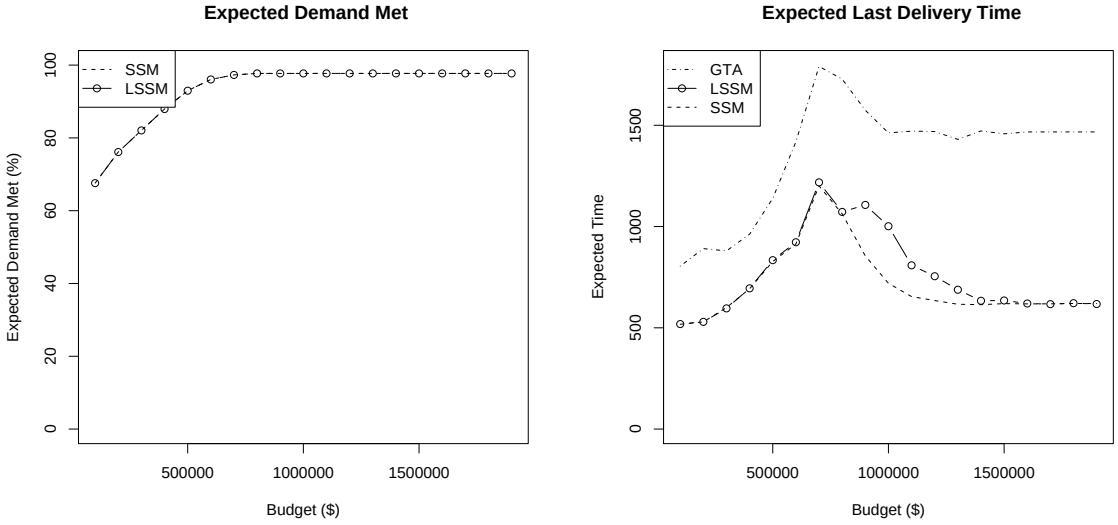

Figure 2.22: Varying the Budget on Benchmark 6 with Decomposition.

Behavioral Analysis of SSSM

Unlike the other storage models, which are easily extended to more complex objective functions, the SSSM can be applied only when the objective function is lexicographic, as is the case here. In this case, the storage decisions for the SSSM are exactly the same as the SSM until the demands are met. Once the demands are satisfied, the SSSM degrades because it cannot determine how to use additional funds to decrease the delivery time. However, as the budget increases, it approaches

the same solution as the SSM because these solutions correspond to storing commodities at all of the repositories. Figure 2.22 presents the experimental results on Benchmark 6, which exhibits this behavior most dramatically (other benchmarks are less pronounced and are omitted). The graph on the left shows how the satisfied demand increases with the budget, while the graph on the right shows how the last delivery time changes. We can see that as the satisfied demand increases the routing time of both algorithms is identical until the total demand is met. At that point, the routing times diverge as the travel distance becomes an important factor in the objective and then re-converge as the budget approaches its maximum and all repositories are storing commodities. These results confirm our expectations and q also demonstrate that the degradation of the decomposed model is not significant when compared to the choices made by the greedy routing algorithm.

2.8 Distribution of Relief Supplies Conclusion

This chapter studied a novel problem in disaster management, the Single Commodity Allocation Problem (SCAP). The SCAP models the strategic planning process for disaster recovery with stochastic last-mile distribution. We proposed a multi-stage stochastic optimization algorithm that yields high-quality solutions to real-world benchmarks provided by Los Alamos National Laboratory. The experimental results on the benchmarks indicate that the algorithm is practical from a computational standpoint and produces significant improvements over existing relief delivery procedures. This work, currently deployed at LANL as part of the National Infrastructure Simulation and Analysis Center (NISAC), is being used to aid government organizations such as the Department of Energy and the Department of Homeland Security in preparing for and responding to disasters.

Chapter 3

Restoration of the Power Network

Restoration of the power grid during the first few days after a major disaster is critical to human welfare since so many services — water pumps, heating systems, and the communication network — rely on it. This chapter solves the following abstract disaster recovery problem: How can power system components be stored throughout a populated area to minimize the blackout size after a disaster has occurred? This case study has only one infrastructure system, the power network. But we will see that just one infrastructure system brings significant computational challenges. The restoration set problem and the restoration order problem from Section 1.3 are essential to solve problems of real-world sizes within the disaster recovery runtime constraints.

This problem is studied in two sections, the stochastic storage problem (Section 3.2) and the restoration routing problem (Section 3.7).

The contributions of this work are:

1. Formalizing the stochastic storage problem and restoration routing problem for power system restoration.
2. An exact mixed-integer programming formulation to the stochastic storage problem (assuming a linearized DC power flow model).
3. A column-generation algorithm that produces near-optimal solutions to the stochastic storage problem under tight time constraints.
4. A multi-stage routing solution for the power system restoration routing problem that produces substantial improvements over field practices on real-life benchmarks of significant sizes.
5. Demonstration that all the solution techniques can bring significant benefits over current field practices.

The rest of this chapter is organized as follows. Section 3.1 reviews related work in power restoration and power system optimization. Section 3.2 presents a specification of the power system stochastic storage problem (PSSSP). Section 3.3 presents an exact MIP formulation using a linearized

DC power flow model. Section 3.4 presents the column-generation algorithm for solving PSSSPs, Section 3.5 presents greedy algorithms for PSSSPs aimed at modeling current practice in the field, and Section 3.6 reports experimental results of the algorithms on some benchmark instances to validate the stochastic storage algorithms. Then Section 3.7 formalizes the restoration routing problem, Section 3.8 presents the multi-stage approach based on constraint injection, and Section 3.9 reports the experimental results validating the approach. Section 3.10 concludes the chapter.

3.1 Related Work

Power engineers have been studying power system restoration (PSR) since at least the 1980s ([1] is a comprehensive collection of work) and the work is still ongoing. The goal of PSR research is to find fast and reliable ways to restore a power system to its normal operational state after a blackout event. This kind of logistics optimization problem is traditionally solved with techniques from industrial engineering and operations research. However, PSR has a number of unique features that prevent the application of traditional optimization methods, including:

1. **Steady-State Behavior:** The flow of electricity over a AC power system is governed by the laws of physics (e.g., Kirchoff's current law and Ohm's law). Hence, evaluating the behavior of the network requires solving a system of nonlinear equations. This can be time-consuming and there is no guarantee that a feasible solution can be found.

2. **Dynamic Behavior:** During the process of modifying the power system's state (e.g., energizing components and changing component parameters), the system is briefly subject to transient states. These short but extreme states may cause unexpected failures [2].

3. **Side Constraints:** Power systems are comprised of many different components, such as generators, transformers, and capacitors. These components have some flexibility in their operational parameters but they may be constrained arbitrarily. For example, generators often have a set of discrete generation levels, and transformers have a continuous but narrow range of tab ratios.

PSR research recognizes that global optimization is an unrealistic goal in such complex nonlinear systems and adopts two main solution strategies. The first is to use domain expert knowledge (i.e. power engineer intuition) to guide an incomplete search of the solution space. These incomplete search methods include *Knowledge-Based Systems* [89], *Expert Systems* [62, 3, 7], and *Local Search* [75, 76]. The second strategy is to approximate the power system with a linear model and solve the approximate problem optimally [108]. Some work has hybridized the two strategies by designing expert systems that solves a series of approximate problems optimally [77, 61].

Interestingly, most of the work in planning PSR has focused on the details of scheduling power system restoration [2, 3]. More specifically, what is the best order of restoration and how should system components be reconfigured during restoration? In fact, these methods assume that all network

components are operational and simply need to be reactivated. In this work, we consider both stockpiling of power network components before a disaster has occurred and recovery operations after a disaster has occurred. This introduces two new decision problems: (1) How to stockpile power-system components in order to restore as much power as possible after a disaster; (2) How to dispatch crews to repair the power-system components in order to restore the power system as quickly as possible. There are strong links between traditional PSR research and our disaster-recovery research. In particular, finding a good order of restoration is central in the repair-dispatching problem. However, the joint repair/recovery problem introduces a combinatorial optimization aspect to restoration that fundamentally changes the nature of the underlying optimization problem. The salient difficulty is to combine two highly complex subproblems, vehicle routing and power restoration, whose objectives may conflict. In particular, the routing aspect optimized in isolation may produce a poor restoration schedule, while an optimized power restoration may produce a poor routing and delay the restoration. To the best of our knowledge, this work is the first PSR application to consider strategic storage decisions and vehicle routing decisions.

The ultimate goal of this work is to mitigate the impact of disasters on multiple infrastructures. Disaster response typically consists of a planning phase, which takes place before the disaster occurs, and a recovery phase, which is initiated after the disaster has occurred. The planning phase often involves a two-stage stochastic or robust optimization problem with explicit scenarios generated by sophisticated weather and fragility simulations (e.g., [30, 31, 102]). The recovery phase is generally a deterministic optimization problem that assumes, to a first approximation, that the damages to the various infrastructures are known. For the power infrastructure, the planning phase — determining where to stockpile power components under various disaster scenarios — is described in Section 3.2, while the recovery phase — how to repair and restore the power infrastructure as fast as possible given the stockpiling decisions — is described in Section 3.7.

3.2 Power System Stochastic Storage

The PSSSP consists of choosing which power system repair components (e.g., components for repairing lines, generators, capacitors, and transformers) to stockpile before a disaster strikes and how those components are allocated to repair the damages of the disaster. A disaster is specified as a set of scenarios, each characterized by a probability and a set of damaged components. In practice, the repository storage constraints may preclude full restoration of the electrical system after a disaster. Therefore, the goal is to select the items to be restored in order to maximize the amount of power served in each disaster scenario prior to external resources being brought in. The primary outputs of a PSSSP are:

1. The amount of components to stockpile before a disaster (first-stage variables);

2. For each scenario, which network items to repair (second-stage variables).

Given:
Power Network: $\mathcal{P}$
Network Item: $i \in N$
Item Type: t_i
Component Types: $t \in T$
Volume: v_t
Storage Locations: $l \in L$
Capacity: c_l
Scenario Data: $s \in S$
Scenario Probability: p_s
Damaged Items: $D_s \subseteq N$
Maximize:
$$\sum_{s \in S} p_s flow_s$$
Subject To:
Storage Capacity
Output:
Which components to store at each location.
Which items to repair in each scenario.

Figure 3.1: The Power System Stochastic Storage Problem (PSSSP) Specification.

Figure 3.1 summarizes the entire problem, which we now describe in detail. The power network is represented as a graph $\mathcal{P} = \langle V, E \rangle$ where V is a set of network nodes (i.e., buses, generators, and loads) and E is a set of the network edges (i.e., lines and transformers). Any of these items may be damaged as a result of a disaster and hence we group them together as a set $N = V \cup E$ of network items. Each network item i has a type t_i that indicates which component (e.g. parts for lines, generators, or buses) is required to repair it. Each component type $t \in T$ in the network has a storage volume v_t. The components can be stored in a set L of warehouses, each warehouse $l \in L$ being specified by a storage capacity c_l. The disaster damages are specified by a set S of different disaster scenarios, each scenario s having a probability of occurring p_s. Each scenario s also specifies its set $D_s \subseteq N$ of damaged items. The objective of the PSSSP is to maximize the expected power flow over all the damage scenarios, where $flow_s$ denotes the power flow in scenario s. In PSSSPs, the power flow is defined to be the amount of watts reaching the load points. The algorithms assume that the power flow for a network $\mathcal{P}$ and a set of damaged items D_s can be calculated by some simulation or optimization model. Note also that the decisions on where to store the components can be tackled in a second step. Once storage quantities are determined, the location aspect of the problem becomes deterministic and can be modeled as a multidimensional knapsack [86].

3.3 The Linear Power Model Approach

PSSSPs can be modeled as two-stage stochastic mixed-integer programming model provided that the power flow constraints for each scenario can be expressed as linear constraints. The linearized DC model, suitably enhanced to capture that some items may be switched on or off, is a natural choice

in this setting, since it has been reasonably successful in optimal transmission switching [44] and network interdiction [38]. Model 8 presents a two-stage stochastic mixed-integer programming model for solving PSSSPs optimally (with a linearized DC power model). The first-stage decision variable x_t denotes the number of stockpiled components of type t. Each second-stage variable is associated with a scenario s. Variable y_{is} specifies whether item i is working, while z_{is} specifies whether item i is operational. Auxiliary variables flow_s denotes the power flow for scenario s, P^l_{is} the real power flow of line i, P^v_{is} the real power flow of node i, and θ_{is} the phase angle of bus i. The objective function (M8.1) maximizes the expected power flow across all the disaster scenarios. Constraint (M8.2) ensures that the stockpiled components do not exceed the storage capacity. Each scenario can repair damaged items only by using the stockpiled components. Constraint (M8.3) ensures that each scenario s uses no more than the stockpiled components of type t. There may be more damaged items of a certain type than the number of stockpiled component of that type, and the optimization model needs to choose which ones to repair, if any. This is captured, for each scenario s, using a linearized DC power flow model (M8.4–M8.13), which extends optimal transmission switching [44] to buses, generators, and loads, since they may be damaged in a disaster. Moreover, since we are interested in a best-case power flow analysis, we assume that generation and load can be dispatched and shed continuously. Constraints (M8.4–M8.7) capture the operational state of the network and specify whether item i is working and/or operational in scenario s. An item is operational only if all buses it is connected to are also operational. Constraint (M8.4) specifies that all undamaged nodes and lines are working. Constraints (M8.5–M8.7) specify which buses (M8.5), load and generators (M8.6), and lines (M8.7) are operational. Constraint (M8.8) computes the total power flow of scenario s in terms of variables P^v_{is}. The conservation of energy is modeled in constraint (M8.9). Constraint (M8.10–M8.11) specify the bounds on the power produced/consumed/transmitted by generators, loads, and lines. Observe that, when these items are not operational, no power is consumed, produced, or transmitted. When a line is non-operational, the effects of Kirchhoff's laws can be ignored, which is captured in the traditional power equations through a big-M transformation [27] (M8.12–M8.13). In this setting, M can be chosen as B_i. Note also that the logical constraints can easily be linearized in terms of the 0/1 variables.

This MIP approach is appealing since it solves PSSSPs optimally for a linearized DC model of power flow. In particular, it provides a sound basis to compare other approaches. However, it does not scale smoothly with the size of the disasters and may be prohibitive computationally in real-life situations. To remedy this limitation, the rest of this paper studies a column-generation inspired approach called configuration-generation.

Model 8 The MIP Model for Power System Stochastic Storage.

Inputs:

$D_{ts} = \{i \in D_s : t_i = t\}$ – demands by type
$V^b = \{i \in N : t_i = bus\}$ – the set of network busses
V_i^g – the set of generators connected to bus i
V_i^l – the set of loads connected to bus i
L – the set of network lines
L_i^- – the from bus of line i
L_i^+ – the to bus of line i
LO_b – the set of exiting lines from bus b
LI_b – the set of entering lines to bus b
B_i – susceptance of line i
$\hat{P}_i^l$ – transmission capacity of line i
$\hat{P}_i^v$ – maximum capacity or load of node i

Variables:

$x_t \in \mathcal{N}$ – number of stockpiled items of type t
$y_{is} \in \{0, 1\}$ – item i is working in scenario s
$z_{is} \in \{0, 1\}$ – item i is operational in scenario s
$flow_s \in \mathcal{R}^+$ – served power for scenario s
$P_{is}^l \in (-\hat{P}_i^l, \hat{P}_i^l)$ – power flow on line i
$P_{is}^v \in (0, \hat{P}_i^v)$ – power flow on node i
$\theta_{is} \in (-\frac{\pi}{6}, \frac{\pi}{6})$ – phase angle on bus i

Maximize:

$$\sum_{s \in S} p_s flow_s \tag{M8.1}$$

Subject To:

$$\sum_{t \in T} v_t x_t \leq \sum_{l} c_l \tag{M8.2}$$

$$\sum_{i \in D_{ts}} y_{is} \leq x_t \qquad \forall t \in T \; \forall s \in S \tag{M8.3}$$

$$y_{is} = 1 \quad \forall i \notin D_s \qquad \forall s \in S \tag{M8.4}$$

$$z_{is} = y_{is} \quad \forall i \in V^b \qquad \forall s \in S \tag{M8.5}$$

$$z_{is} = y_{is} \wedge y_{js} \qquad \forall j \in V^b \; \forall i \in V_j^g \cup V_j^l \; \forall s \in S \tag{M8.6}$$

$$z_{is} = y_{is} \wedge y_{L_i^+ s} \wedge y_{L_i^- s} \qquad \forall i \in L \; \forall s \in S \tag{M8.7}$$

$$flow_s = \sum_{i \in V^b} \sum_{j \in V_i^l} P_{js}^v \qquad \forall s \in S \tag{M8.8}$$

$$\sum_{j \in V_i^l} P_{js}^v = \sum_{j \in V_i^g} P_{js}^v + \sum_{j \in LI_i} P_{js}^l - \sum_{j \in LO_i} P_{js}^l \qquad \forall i \in V^b \; \forall s \in S \tag{M8.9}$$

$$0 \leq P_{is}^v \leq \hat{P}_i^v z_{is} \qquad \forall i \in V_j^g \cup V_j^l \; \forall s \in S \tag{M8.10}$$

$$-\hat{P}_{is}^l z_{is} \leq P_{is}^l \leq \hat{P}_{is}^l z_{is} \qquad \forall i \in L \; \forall s \in S \tag{M8.11}$$

$$P_{is}^l \leq B_i(\theta_{L_i^+ s} - \theta_{L_i^- s}) + M(\neg z_{is}) \qquad \forall i \in L \; \forall s \in S \tag{M8.12}$$

$$P_{is}^l \geq B_i(\theta_{L_i^+ s} - \theta_{L_i^- s}) - M(\neg z_{is}) \qquad \forall i \in L \; \forall s \in S \tag{M8.13}$$

Model 9 The PSSSP Configuration-Generation Master Problem.

Inputs:

$\mathcal{W}$ – set of configurations
w_t – amount of components of type t in $w \in W$
flow_{ws} – served power for w in scenario s

Variables:

$x_t \in \mathcal{N}$ – number of stockpiled items of type t
$y_{ws} \in \{0,1\}$ – 1 if configuration w is used in s
$flow_s \in \mathcal{R}^+$ – power flow for scenario s

Maximize:

$$\sum_{s \in S} p_s flow_s \qquad \text{(M9.1)}$$

Subject To:

$$\sum_{t \in T} v_t x_t \leq \sum_{l} c_l \qquad \text{(M9.2)}$$

$$\sum_{w \in W} w_t y_{ws} \leq x_t \qquad \forall s \in S \ \ \forall t \in T \qquad \text{(M9.3)}$$

$$\sum_{w \in W} y_{ws} = 1 \qquad \forall s \in S \qquad \text{(M9.4)}$$

$$\sum_{w \in W} \mathit{flow}_{ws} y_{ws} = flow_s \qquad \forall s \in S \qquad \text{(M9.5)}$$

3.4 A Configuration-Generation Approach

When the optimal values of the first-stage variables are known, the PSSSP reduces to solving a restoration problem for each scenario s, i.e., maximizing the power flow for scenario s under the stored resources specified by the first-stage variables.

The configuration-generation approach takes a dual approach: It aims at combining feasible solutions of each scenario to obtain high-quality values for the first-stage variables. In this setting, a configuration is a tuple $w = \langle w_1, \ldots, w_k \rangle$ ($T = \{1, \ldots, k\}$), where w_t specifies the number of items of type t being stockpiled. A configuration is feasible if it satisfies the storage constraints, i.e.,

$$\sum_{t \in T} v_t w_t \leq \sum_{l \in L} c_l.$$

For each scenario s, the optimal power flow of a feasible configuration w is denoted by flow_{ws}. Once a set of feasible configurations is available, a mixed-integer program (the Master Problem) selects a set of configurations, one per scenario, maximizing the expected power flow across all scenarios. Model 9 presents the MIP model. In the model, the objective (M9.1) specifies that the goal is to maximize the expected flow. Constraint (M9.2) enforces the storage requirements, while constraint (M9.3) links the number of components x_t of type t used by scenario s with the configuration variables y_{ws}. Constraint (M9.4) specifies that each scenario uses exactly one configuration. Constraint (M9.5) computes the power flow of scenario s using the configuration variables y_{ws}.

It is obviously impractical to generate all configurations for all scenarios. As a result, we follow a configuration-generation approach in which configurations are generated on demand to improve

Model 10 The Generic Configuration-Generation Subproblem for a Scenario.

Let:

s – a scenario

$T_t = \{i \in N : t_i = t\}$ – the set of nodes of type t

Variables:

$r_i \in \{0, 1\}$ – 1 if item i is repaired

$flow \in \mathcal{R}^+$ – served power

Maximize:

$(flow, -\sum_i r_i)$ (M10.1)

Subject To:

$storageConstraint(\{T_t\}_{t \in T}, \{r_i\}_{i \in N})$ (M10.2)

$flowConstraints(\mathcal{P}, D_s, \{r_i\}_{i \in N}, flow)$ (M10.3)

the objective of the Master Problem. Recall that our goal is twofold: First, we aim at designing an approach that is largely independent of the power flow simulation or optimization model. Second, we aim at producing an optimization model that scales to large disasters. For these reasons, we generate configurations based on techniques inspired by online stochastic combinatorial optimization [101, 103].

3.4.1 The Configuration-Generation Subproblem

In our configuration-generation approach, the configurations are generated for each scenario independently and in a systematic fashion. Figure 10 describes a generic optimization model for generating a configuration. In the model, the storage and power flow constraints are abstracted for generality. The storage constraints contain at least the storage constraints of the PSSSP but generally impose additional constraints to generate "interesting" configurations for the Master Problem. The power flow constraints are abstracted to make the approach independent of the power flow model. The optimization model features a lexicographic objective (M10.1), seeking first to maximize the power flow and then to minimize the number of repaired components. The storage constraints (M10.2) are expressed in terms of the decision variables r_i that indicate whether item i is repaired; they are described shortly. Constraint (M10.3) encapsulates the power flow model that computes the flow from the values of the decision variables.

The configuration-generation algorithm generates two fundamental types of configurations for a given scenario. Each type takes into account information from the other scenarios and instantiates the generic model in Model 10 with specific storage constraints.

Upward Configurations The intuition behind upward configurations is as follows. Some storage decisions coming from other scenarios are fixed by a configuration w and the goal is to generate the best configuration possible for scenario s given these decisions, i.e., a configuration for scenario s that maximizes its power flow given that the repairs must include at least w_t components of type t. More precisely, *upwardConfiguration(w,s)* denotes the solution to the model in Figure 10 where the

storage constraint becomes

$$\sum_{t \in T} v_t \max(w_t, \sum_{i \in T_t} r_i) \leq \sum_{l} c_l.$$

Downward Configurations The intuition behind downward configurations is as follows. Our goal is to give an opportunity for a configuration w from another scenario to be selected in the Master Problem while ensuring that all other scenarios generate a mutually compatible configuration. As a result, we seek to maximize the power flow for scenario s, while not exceeding the storage requirements of w. More precisely, *downwardConfiguration(w,s)* denotes the solution to the model in Figure 10 where the storage constraint becomes

$$\forall t \in T : \sum_{i \in T_t} r_i \leq w_t.$$

Two special cases of upward and downward configurations are important for initializing the configuration-generation process: the clairvoyant and the no-repair configurations.

Clairvoyant Configurations The clairvoyant configuration for a scenario s is simply the upward configuration with no storage constraint imposed:

$$\textit{clairvoyant}(s) = \textit{upwardConfiguration}(\langle 0, \ldots, 0 \rangle, s).$$

Scenarios whose clairvoyant configuration is selected in the Master Problem cannot be improved.

No-Repair Configurations The no-repair configuration for a scenario s is simply the downward configuration with no storage availability:

$$\textit{no-repair}(s) = \textit{downwardConfiguration}(\langle 0, \ldots, 0 \rangle, s).$$

The no-repair configuration guarantees that each scenario can select at least one configuration in the Master problem.

3.4.2 The Configuration-Generation Algorithm

Figure 3.2 presents the complete configuration-generation algorithm. Lines 1–4 describe the initialization process, while lines 7–12 specify how to generate new configurations. The overall algorithm terminates when the newly generated configurations do not improve the quality of the Master Problem.

Initialization The initialization step generates the initial set of configurations. These contain the clairvoyant and the no-repair configurations for each scenario, as well as the downward configurations obtained from the expected clairvoyant, i.e., the configuration w^e whose element t is defined by

$$w^e_t = \sum_{s \in S} p_s \, \textit{clairvoyant}(s).$$

The Configuration-Generation Process At each iteration, the algorithm solves the Master Problem. If the solution has not improved over the previous iteration, the algorithm completes and returns its configuration, i.e., the values of the decision variables x_t describing how many elements of component type t must be stockpiled.

Otherwise, the algorithm considers each scenario s in isolation. The key idea is to solve the Master Problem without scenario s, which we call a restricted Master Problem, and to derive new configurations from its solution w^{-s}. In particular, for each scenario s, the configuration-generation algorithm generates

1. one upward configuration for s (line 9);
2. one downward configuration for every other scenario in $S \setminus \{s\}$ (line 12).

The upward configuration for s is simply the best possible configuration given the decisions in the restricted Master Problem, i.e.,

$$\mathit{upwardConfiguration}(w^{-s}, s)$$

The downward configurations for the other scenarios are obtained by selecting an existing configuration w^{+s} for s that if added to the restricted Master solution, would violate the storage constraint. Downward configurations of the form

$$\mathit{downwardConfiguration}(w^{+s}, j)$$

are then computed for each scenario $j \in S \setminus \{s\}$ in order to give the Master an opportunity to select w^{+s}. The configuration w^{+s} aims at being desirable for s, while taking into account the requirement of the other scenarios. Initially, w^{+s} is the clairvoyant solution. In general, w^{+s} is the configuration in $\mathcal{W}$ that, when scaled to satisfy the storage constraints, maximizes the power flow for scenario s, i.e.,

$$\begin{array}{ll} \max_{w \in \mathcal{W}} & flow_w p \\ \text{subject to} & \\ & \exists t \in T : w_t > w_t^{-s} \\ & \forall t \in T : w_t p \leq w_t^{-s} \\ & 0 \leq p \leq 1 \end{array}$$

Simulation-Independent Optimization The configuration-generation algorithm relies only on the power flow model in the subproblem of Figure 10. This is essentially an optimal transmission switching model with limits on the component types that can be repaired. It can be solved in various ways, making the overall approach independent of the power flow model.

3.5 A Greedy Storage Model

This section presents a basic greedy storage model that emulates standard practice in storage procedures and provides a baseline for evaluating our optimization algorithms. However, to the best of

```
ConfigurationGeneration()
 1  W ← {no-repair(s) | s ∈ S}
 2  W ← W ∪ clairvoyant(s) | s ∈ S}
 3  w^e ← expectedClairvoyant(S)
 4  W ← W ∪ {downwardConfiguration(w^e, s) | s ∈ S}
 5  while The master objective is increasing
 6  do w^m ← Master(S)
 7     for s ∈ S
 8     do w^{-s} ← Master(S \ {s})
 9        W ← W ∪ upwardConfiguration(w^{-s}, s)
10        w^{+s} ← selectConfiguration(w^{-s}, s)
11        for j ∈ S \ {s}
12        do W ← W ∪ downwardConfiguration(w^{+s}, j)
13  return w^m
```

Figure 3.2: The Configuration-Generation Algorithm for the PSSSP.

my knowledge, the storage models used in practice are rather ad hoc or based on past experience and the models presented in this section probably improve current procedures. In fact, it is well known that power companies often rely on spare parts from neighboring regions in post-disaster situations.

The greedy storage algorithm generates a storage configuration w^h by computing first a distribution of the component types and then filling the available storage capacity with components to match this distribution. Once a scenario s is revealed, the quality of greedy storage can be evaluated by computing a downward configuration

$$downwardConfiguration(w^h, s).$$

The distribution used to produce w^h is based on the number of occurrences of the component types in the undamaged network. This metric is meaningful because it models the assumption that every component type is equally likely to be damaged in a disaster. The computation for a distribution Pr proceeds as follows. When all of the components have a uniform size v, the quantity of component type i is

$$\lfloor (\sum_{l \in L} c_l Pr(i))/v \rfloor.$$

When the component types have different sizes, the storage configuration is the solution of the optimization problem

$$\begin{aligned} \min_{t \in T} \quad & |\frac{w_t^h}{\sum_{i \in T} w_i^h} - Pr(i)| \\ \text{subject to} \quad & \\ & \sum_{t \in T} w_t^h v_t \leq \sum_{l \in L} c_l \\ & 0 \leq w_i^h. \end{aligned}$$

Benchmark	$\|N\|$	$\|S\|$	$\max_{s \in S}(\|D_s\|)$
BM1	326	3	22
BM3	266	18	61
BM4	326	18	121
BM5	1789	4	255

Table 3.1: Features of the PSSSP Benchmarks.

3.6 Benchmarks and Results

This section reports the experimental results of the proposed models. It starts by describing the benchmarks and the algorithms. It then presents the quality results and the efficiency results. It concludes by reporting a variety of statistics on the behavior of the configuration-generation algorithms.

3.6.1 Benchmarks

The benchmarks were produced by Los Alamos National Laboratory and are based on the electrical infrastructure of the United States. The disaster scenarios were generated by state-of-the-art hurricane simulation tools similar to those used by the National Hurricane Center [4, 88]. Their sizes are presented in Table 3.1, giving the number of network items, the number of scenarios, and the size of the largest damage produced by the scenarios. For instance, BM5 considers a network with 1789 items and four disaster scenarios. The worst disaster damages 255 items in the network. Each benchmark is evaluated for a large variety of storage capacities (i.e., different values of c_l). This makes it possible to evaluate the behavior of the algorithms under a wide variety of circumstances, as well as to study the tradeoff between storage (or budget) and the quality of the restoration. Overall, more than 70 configurations of each algorithm were studied in these experiments.

3.6.2 Implementation of the Algorithms

The optimization algorithms were implemented in the COMET system [43, 73, 100] and the experiments were run on Intel Xeon CPU 2.80 GHz machines running 64-bit Linux Debian. The experiments use the standard linearized DC power flow equations for the subproblems. To quantify its benefits compared to field practice, the configuration-generation algorithm is compared to the greedy storage procedures presented earlier. The global quality of the configuration-generation algorithm is evaluated by comparison to the optimal MIP model presented in Section 3.3 and to the clairvoyant solution, i.e., the solution obtained when the actual disaster scenario is known in advance. Benchmarks 1 and 3 are small enough that the MIP models can be solved to optimality. However, Benchmarks 4 and 5 present large damage scenarios with over 100 damaged components, and optimal solutions to the MIP models may not be obtained in a reasonable amount of time. The following time limits are then imposed: 18 hours for the global MIP, two hours for the evaluation of each greedy and clairvoyant scenario, and 15 minutes for configuration-generation subproblems.

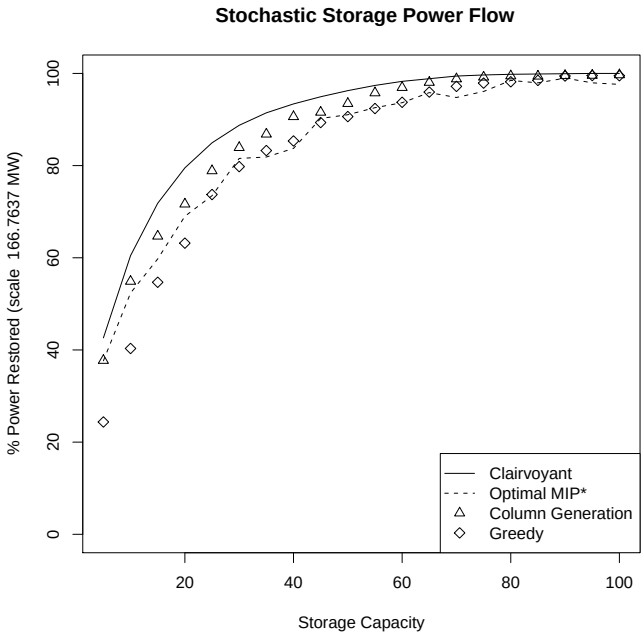

Figure 3.3: Solution Quality for the PSSSP Algorithms on Benchmark 4.

Benchmark	Clairvoyant	MIP Model	Configuration Generation	Greedy
BM1	100%	99.1%	98.5%	73.4%
BM3	100%	97.4%	96.6%	71.8%
BM4	100%	91.4%	97.5%	88.3%
BM5	100%	77.5%	85.4%	69.3%

Table 3.2: Experimental Results on Aggregated Solution Quality.

3.6.3 Quality of the Algorithms

Figure 3.3 depicts the quality results for Benchmark 4, i.e., the percentage of the blackout restored by the various algorithms for a given storage capacity. The figure shows the results for the greedy procedure, the configuration-generation algorithm, the MIP model, and the expected clairvoyant solution.

The results indicate that the MIP model and the configuration-generation algorithm produce significant improvements over the greedy approach, reducing the blackout by almost 20% in the best case. The improvements are especially significant for low storage capacities, which seem to be the case in practice. Remarkably, the configuration-generation algorithm outperforms the MIP model, which always reaches its time limit. On this set of benchmarks, the MIP model and the configuration-generation algorithm are close to the clairvoyant solution, indicating that the stochasticity in PSSSPs is reasonable. This implies that our algorithms produce high-quality solutions for the scenarios.

Space constraints prevent including similar graphs for all benchmark classes. Instead, we aggregate these results for each benchmark class, averaging the quality over 20 storage capacities for each benchmark. The results are presented in Table 3.2, which depicts the average relative gap of

Benchmark	Clairvoyant	MIP Model	Configuration Generation	Greedy
BM1	100%	97.9%	96.3%	66.0%
BM3	100%	92.8%	92.8%	21.9%
BM4	100%	62.7%	75.0%	34.2%
BM5	100%	70.7%	81.5%	11.4%

Table 3.3: Aggregated Solution Quality on Small Storage Capacities.

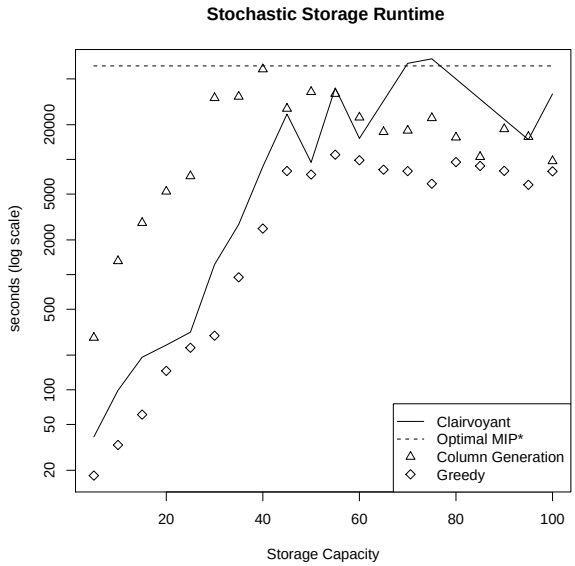

Figure 3.4: Experimental Results for the PSSSP Algorithm on Benchmark 4.

each algorithm from the clairvoyant solution. The configuration-generation approach brings substantial benefits over the greedy approach in smaller benchmarks and over both the greedy and MIP approach on larger ones, demonstrating its scalability. Table 3.3 gives another perspective: it aggregates results only for small storage capacities, which often correspond to real-life situations. The benefits of the configuration-generation approach over the greedy algorithm are dramatic and the gain over the MIP is even more significant on the larger benchmarks.

3.6.4 Performance of the Algorithms

Figure 3.4 depicts the performance results for Benchmark 4 under 20 different storage capacities. It shows the run time of the MIP Model and the configuration-generation algorithm in seconds, using a log scale given the significant performance difference between the algorithms. The results show substantial performance improvements for the configuration-generation algorithm over the MIP model. This is especially the case for low to medium storage capacities, typical in the field. For these storage configurations, the configuration-generation algorithm runs from 2 to 100 times faster than the MIP model (despite its time limit) and is about three times faster in average. In summary, on large instances, the configuration-generation approach improves both the quality and performance of the MIP models. Table 3.4 presents the average run time results for all benchmarks and also

Benchmark	Clairvoyant	MIP Model	Configuration Generation	Greedy
BM1	2	6	8.84	2
BM3	18	21351	415	18
BM4	20551	64807	20038	5133
BM5	20544	63704	21349	8308

Table 3.4: PSSSP Benchmark Runtime (Seconds).

Benchmark	Clairvoyant	MIP Model	Configuration Generation	Greedy
BM1	2	4	10	2
BM3	12	13060	179	12
BM4	178	64807	3366	98
BM5	14613	60502	10371	1123

Table 3.5: Benchmark Runtime on Small Storage Capacities (Seconds).

gives the time to evaluate the greedy and expected clairvoyant problems as a basis for comparison. These results are quite interesting: the MIP model behaves very well for reasonably small disasters, but does not scale well to larger ones. Since our goal is to tackle disaster planning and restoration at the state level, we see a fundamental limitation of a pure MIP approach. It is also interesting to observe that the configuration-generation algorithm takes about four times as long for evaluating the greedy power flow. This is quite remarkable and indicates that the configuration-generation is likely to scale well to even larger problem sizes. Overall, these results indicate that the configuration generation algorithm produces near-optimal solutions under tight time constraints and represents an appealing approach to PSSSPs.

3.6.5 Behavior of the Configuration-Generation Algorithm

It is useful to study the behavior of the configuration-generation algorithm in order to understand its performance. Table 3.6 reports, in average, the number of configurations generated, as well as the number of downward and upward configurations in the final solution. The results indicate that the algorithm needs only a few configurations to produce a high-quality solution. Recall that Benchmark 3 and Benchmark 4 have 18 scenarios, which means that the algorithm generates on average fewer than seven configurations per scenario. The types of configurations are relatively well balanced, although there are clearly more upward configurations on average. This is partly caused by the variance in the damage scenarios: as the storage capacity increases more of the small-damage scenarios can select their clairvoyant solution. Figure 3.5 details these results for Benchmark 4.

3.7 Power Restoration Vehicle Routing Problem

This section formalizes the Power Restoration Vehicle Routing Problem (PRVRP).

Benchmark	Configurations	Downward Config.	Upward Config.
BM1	7.8	0.9	2.1
BM3	115.5	8.9	9.1
BM4	136.50	7.75	10.25
BM5	30.42	1.89	2.11

Table 3.6: The Behavior of the Configuration Generation for PSSSPs.

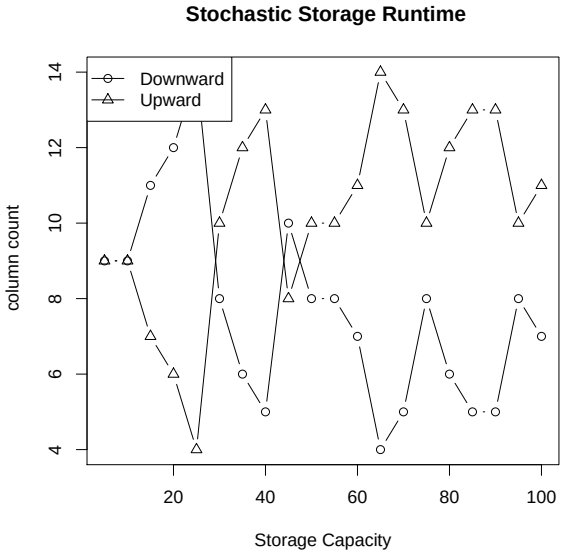

Figure 3.5: The Types of Configurations in the Final PSSSP Solutions.

3.7.1 The Routing Component

The PRVRP is defined in terms of a graph $G = \langle S, E \rangle$ where S represents sites of interest and E are the travel times between sites. The sites are of four types: (1) the depots H^+ from which repair vehicles depart; (2) the depots H^- to which repair vehicles must return; (3) the depots W^+ where stockpiled resources are located; and (4) the locations W^- where electrical components (e.g., lines, buses, and generators) must be repaired. Due to infrastructure damages, the travel times on the edges are typically not Euclidian, but they do form a metric space. For simplicity, this work assumes that the graph is complete and $t_{i,j}$ denotes the distance between sites i and j.

The restoration has at its disposal a set V of vehicles. Each vehicle $v \in V$ is characterized by its departure depot h_v^+, its returning depot h_v^-, and its capacity c_v. Vehicle v starts from h_v^+, performs a number of repairs, and return to h_v^-. It cannot carry more resources than its capacity.

The restoration must complete a set J of restoration jobs. Each job j is characterized by a pickup location p_j^+, a repair location p_j^-, a volume d_j, a service time s_j, and a network item n_j. Performing a job consists of picking up repair supplies at p_j^+ (which uses d_j units of the vehicle's capacity), traveling to site p_j^-, and repairing network item n_j at p_j^- for a duration s_j. After completion of job j, network item n_j is working and can be activated.

A solution to the PRVRP associates a route $\langle h_v^+, w_1, \ldots, w_k, h_v^- \rangle$ with each vehicle $v \in V$ such

that all locations are visited exactly once. A solution can then be viewed as assigning to each location $l \in H^+ \cup W^+ \cup W^-$, the vehicle $vehicle(l)$ visiting l, the load $load_l$ of the vehicle when visiting l, the next destination of the vehicle (i.e., the successor σ_l of l in the route of l), and the earliest arrival time EAT_l of the vehicle at location l. The loads at the sites can be defined recursively as follows:

$$\begin{array}{lcll} load_l & = & 0 & \text{if } l \in H^+ \\ load_{\sigma_l} & = & load_l + d_l & \text{if } l \in W^+ \\ load_{\sigma_l} & = & load_l - d_l & \text{if } l \in W^-. \end{array}$$

Pickup locations increase the load, while delivery locations decrease the load. The earliest arrival times can be defined recursively as

$$\begin{array}{lcll} EAT_l & = & 0 & \text{if } l \in H^+ \\ EAT_{\sigma_l} & = & EAT_l + t_{l,\sigma_l} & \text{if } l \in W^+ \\ EAT_{\sigma_l} & = & EAT_l + t_{l,\sigma_l} + s_l & \text{if } l \in W^-. \end{array}$$

The earliest arrival time of a location is the earliest arrival time of its predecessor plus the travel time and the service time for repair locations. The earliest departure time EDT_l from a location is simply the earliest arrival time (to which the service time is added for delivery locations). A solution must satisfy the following constraints:

$$\begin{array}{lcll} vehicle(p_j^+) & = & vehicle(p_j^-) & \forall j \in J \\ EAT_{p_j^+} & < & EAT_{p_j^-} & \forall j \in J \\ load_l & \leq & c_{vehicle(l)} & \forall l \in W^+ \cup W^-. \end{array}$$

The first constraint specifies that the same vehicle performs the pairs of pickups and deliveries, the second constraint ensures that a delivery takes place after its pickup, while the third constraint ensures that the capacities of the vehicles are never exceeded.

The power network $\mathcal{PN} = \langle N, L \rangle$ is defined in terms of a set N of nodes and a set L of lines. The nodes $N = N^b \cup N^g \cup N^l$ are of three types: the buses N^b, the generators N^g, and the loads N^l. Each bus b is characterized by its set N_b^g of generators, its set N_b^l of loads, its set LO_b of exiting lines, and its set LI_b of entering lines. The maximum capacity or load of a node in $N^g \cup N^l$ is denoted by $\hat{P}_i^v$. Each line l is characterized by its susceptance B_l and its transmission capacity $\hat{P}_l^l$. Its from-bus is denoted by L_l^- and its to-bus by L_l^+. The network item n_j of job j is an item from $N \cup L$. The set $\{n_j \mid j \in J\}$ denotes the damaged items D.

The PRVRP objective is to minimize the total watts/hours of blackout, i.e., $\int unservedLoad(t)\, dt$. Each repair job provides an opportunity to reduce the blackout area (e.g., by bringing a generator up) and the repairs occur at discrete times $T_1 \leq T_2 \leq \ldots \leq T_{|J|}$. Hence the objective can be rewritten as the minimization of $\sum_{i=2}^{|J|} unservedLoad(T_{i-1}) \times (T_i - T_{i-1})$; the precise meaning of "unserved load" in this formula will be presented shortly. At each discrete time T_i, exactly i network elements have been repaired and can be activated, but it may not be beneficial to reactivate all of them. Hence, since we are interested in a best-case power flow analysis, we assume that, after each repair, the optimal set of elements is activated to serve as much of the load as possible. Generation and load can also be dispatched and shed appropriately.

Under these assumptions, computing the unserved load becomes an optimization problem in itself. Model 11 depicts a MIP model for minimizing the unserved load assuming a linearized DC model of power flow. The inputs of the model are the power network (with the notations presented earlier), the set D of damaged nodes, the set R of already repaired nodes, and the value *MaxFlow* denoting the maximum power when all items are repaired. Variable y_i captures the main decision in the model, i.e., whether to reactivate repaired item i. Auxiliary variable z_i determines if item i is operational. The remaining decision variables determine the power flow on the lines, loads, and generators, as well as the phase angles for the buses. The model objective minimizes the unserved load. Constraints (M11.2–M11.6) determine which items can be activated and which are operational. Constraints (M11.2) specify that undamaged items are activated and constraints (M11.3) specify that damaged items cannot be activated if they have not been repaired yet. Constraints (M11.4–M11.6) describe which items are operational. An item is operational only if all buses it is connected to are operational. Constraints (M11.4) consider the buses, constraints (M11.5) the loads and generators that are connected to only one bus, and constraints (M11.6) the lines that are connected to two buses. Constraints (M11.7) express Kirchhoff's Current Law, while constraints (M11.8–M11.11) impose restrictions on power flow, consumption, and production. Constraints (M11.8) impose lower and upper bounds on the power consumption and production for loads and generators and ensure that a non-operational load or generator cannot consume or produce power. Constraints (M11.9) impose similar bounds on the lines. Finally, constraints (M11.10–M11.11) define the flow on the lines in terms of their susceptances and the phase angles. These constraints are ignored when the line is non-operational through a big-M transformation. In practice, M can be set to B_i and the logical connectives can be transformed into linear constraints over 0/1 variables.

The PRVRP is extremely challenging from a computational standpoint, since it composes two subproblems that are challenging in their own right. On the one hand, pickup and delivery vehicle-routing problems have been studied for a long time in operations research. For reasonable sizes, they are rarely solved to optimality. In particular, when the objective is to minimize the average delivery time (which is closely related to the PRVRP objective), Campbell et al. [23] have shown that MIP approaches have serious scalability issues. The combination of constraint programming and large-neighborhood search has been shown to be very effective in practice and has the advantage of being flexible in accommodating side constraints. On the other hand, computing the unserved load generalizes optimal transmission switching, which has also been shown to be challenging for MIP solvers [44]. In addition to line switching, the PRVRP also considers the activation of load and generators. Therefore, it is highly unlikely that a direct approach, combining MIP models for both the routing and power flow subproblems, would scale to the size of even small restorations. Our experimental results with such an approach were in fact very discouraging. The rest of this chapter presents an approach that aims at decoupling both subproblems as much as possible while still producing high-quality routing schedules.

Model 11 Minimizing Unserved Load in a Power Network.

Inputs:

$\mathcal{PN} = \langle N, L \rangle$	– the power network
D	– the set of damaged items
R	– the set of repaired items
$MaxFlow$	– the maximum flow

Variables:

$y_i \in \{0, 1\}$	– item i is activated
$z_i \in \{0, 1\}$	– item i is operational
$P_i^l \in (-\hat{P}_i^l, \hat{P}_i^l)$	– power flow on line i
$P_i^v \in (0, \hat{P}_i^v)$	– power flow on node i
$\theta_i \in (-\frac{\pi}{6}, \frac{\pi}{6})$	– phase angle on bus i

Minimize:

$$MaxFlow - \sum_{b \in N^b} \sum_{i \in N_b^l} P_i^v \tag{M11.1}$$

Subject To:

$$y_i = 1 \qquad \forall i \in (N \cup L) \setminus D \tag{M11.2}$$

$$y_i = 0 \qquad \forall i \in D \setminus R \tag{M11.3}$$

$$z_i = y_i \qquad \forall i \in N^b \tag{M11.4}$$

$$z_i = y_i \wedge y_j \qquad \forall j \in N^b \; \forall i \in N_j^g \cup N_j^l \tag{M11.5}$$

$$z_i = y_i \wedge y_{L_i^+} \wedge y_{L_i^-} \qquad \forall i \in L \tag{M11.6}$$

$$\sum_{j \in N_i^l} P_j^v = \sum_{j \in N_i^g} P_j^v + \sum_{j \in LI_i} P_j^l - \sum_{j \in LO_i} P_j^l \qquad \forall i \in N^b \tag{M11.7}$$

$$0 \leq P_i^v \leq \hat{P}_i^v z_i \qquad \forall j \in N^b \; \forall i \in N_j^g \cup N_j^l \tag{M11.8}$$

$$-\hat{P}_i^l z_i \leq P_i^l \leq \hat{P}_i^l z_i \qquad \forall i \in L \tag{M11.9}$$

$$P_i^l \geq B_i(\theta_{L_i^+} - \theta_{L_i^-}) + M(\neg z_i) \qquad \forall i \in L \tag{M11.10}$$

$$P_i^l \leq B_i(\theta_{L_i^+} - \theta_{L_i^-}) - M(\neg z_i) \qquad \forall i \in L \tag{M11.11}$$

3.8 Constraint Injection

As mentioned, a direct integration of the routing and power-flow models, where the power-flow model is called upon to evaluate the quality of (partial) routing solutions, cannot meet the real-time constraints imposed by disaster recovery. For this reason, we explore a multi-stage approach exploiting the idea of *constraint injection*. Constraint injection enables a decoupling of the routing and power-flow models, while capturing the restoration aspects in the routing component. It exploits two properties to perform this decoupling. First, once all the power has been restored, the subsequent repairs do not affect the objective and the focus can be on the routing aspects only. Second, and most importantly, a good restoration schedule can be characterized by a partial ordering on the repairs. As a result, the key insight behind constraint injection is to impose, on the routing subproblem, precedence constraints on the repair crew visits that capture good restoration schedules.

The injected constraints are obtained through two joint optimization/simulation problems. First, the Minimum Restoration Set Problem computes the smallest set of items needed to restore the grid to full capacity. Then, the Restoration Order Problem determines the optimal (partial) order for

MULTI-STAGE-PRVRP(*Network* $\mathcal{PN}$, *PRVRP* G)
1 $\mathcal{S} \leftarrow MinimumRestorationSetProblem(\mathrm{G}, \mathcal{PN})$
2 $\mathcal{O} \leftarrow RestorationOrderProblem(\mathcal{PN}, \mathcal{S})$
3 $\mathcal{R} \leftarrow PrecedenceRoutingProblem(\mathrm{G}, \mathcal{O})$
4 **return** $PrecedenceRelaxation(\mathcal{PN}, \mathcal{R})$

Figure 3.6: The Multi-Stage PRVRP Algorithm.

Model 12 Minimum Restoration Set Problem (MRSP).

Inputs:

$\mathcal{PN} = \langle N, L \rangle$	– the power network
D	– the set of damaged items
MaxFlow	– the maximum flow

Variables:

$y_i \in \{0, 1\}$	– item i is activated
$z_i \in \{0, 1\}$	– item i is operational
$P_i^l \in (-\hat{P}_i^l, \hat{P}_i^l)$	– power flow on line i
$P_i^v \in (0, \hat{P}_i^v)$	– power flow on node i
$\theta_i \in (-\frac{\pi}{6}, \frac{\pi}{6})$	– phase angle on bus i

Minimize:

$$\sum_{i \in N \cup L} y_i \qquad \text{(M12.1)}$$

Subject To:

$$\sum_{b \in N^b} \sum_{i \in N_b^l} P_i^v = MaxFlow \qquad \text{(M12.2)}$$

$$y_i = 1 \quad \forall i \in N \setminus D \qquad \text{(M12.3)}$$

Constraints (M11.4–M11.11) from Model 11

restoring the selected subset in order to minimize the total blackout hours. The resulting partial order provides the precedence constraints injected in the pickup and delivery vehicle-routing optimization. Once the routing solution is obtained, injected precedence constraints between vehicles are relaxed, since they may force vehicles to wait unnecessarily. The final algorithm is a multi-stage joint optimization/simulation algorithm depicted in Figure 3.6. Each of the steps is now reviewed in detail.

3.8.1 The Minimum Restoration Set Problem

(MRSP) determines the smallest set of items to restore for ensuring full network capacity. Model 12 depicts the mathematical model using a linear DC model. The optimization is closely related to the model for the unserved load presented in Model 11, but has three significant changes. First, the objective (M12.1) now minimizes the number of repairs. Second, constraint (M12.2) ensures that the network will operate at full capacity. The remaining model constraints are identical to (M11.4–M11.11) in Model 11. However, constraints (M11.3) from Model 11 is excluded since we allow all items to be repaired.

Model 13 Restoration Ordering Problem (ROP).

Inputs:

$\mathcal{PN} = \langle N, L \rangle$	– the power network
D	– the set of damaged items
R	– the set of items to repair
$MaxFlow$	– the maximum flow

Variables:

$flow_k$	– the flow in step k
$o_{ik} \in \{0, 1\}$	– item i is repaired in step k
$y_{ik} \in \{0, 1\}$	– item i is activated in step k
$z_{ik} \in \{0, 1\}$	– item i is operational in step k
$P^l_{ik} \in (-\hat{P}^l_i, \hat{P}^l_i)$	– power flow on line i in step k
$P^v_{ik} \in (0, \hat{P}^v_i)$	– power flow on node i in step k
$\theta_{ik} \in (-\frac{\pi}{6}, \frac{\pi}{6})$	– phase angle on bus i in step k

Minimize

$$\sum_{k=1}^{|R|} (MaxFlow - flow_k) \tag{M13.1}$$

Subject To: $(1 \leq k \leq |R|)$

$$flow_k = \sum_{b \in N^b} \sum_{i \in N^l_b} P^v_{ik} \tag{M13.2}$$

$$\sum_{r \in R} o_{rk} = k \tag{M13.3}$$

$$o_{rk-1} \leq o_{rk} \qquad \forall r \in R \tag{M13.4}$$

$$y_{ik} \leq o_{ik} \qquad \forall i \in D \tag{M13.5}$$

$$y_{ik} = 1 \qquad \forall i \in (N \cup L) \setminus D \tag{M13.6}$$

$$y_{ik} = 0 \qquad \forall i \in D \setminus R \tag{M13.7}$$

$$z_{ik} = y_{ik} \qquad \forall i \in N^b \tag{M13.8}$$

$$z_{ik} = y_{ik} \wedge y_{jk} \qquad \forall j \in N^b \; \forall i \in N^g_j \cup N^l_j \tag{M13.9}$$

$$z_{ik} = y_{ik} \wedge y_{L^+_i k} \wedge y_{L^-_i k} \qquad \forall i \in L \tag{M13.10}$$

$$\sum_{j \in N^l_i} P^v_{jk} = \sum_{j \in N^g_i} P^v_{jk} + \sum_{j \in LI_i} P^l_{jk} - \sum_{j \in LO_i} P^l_{jk} \qquad \forall i \in N^b \tag{M13.11}$$

$$0 \leq P^v_{ik} \leq \hat{P}^v_i z_{ik} \qquad \forall j \in N^b \; \forall i \in N^g_j \cup N^l_j \tag{M13.12}$$

$$-\hat{P}^l_i z_{ik} \leq P^l_i \leq \hat{P}^l_i z_{ik} \qquad \forall i \in L \tag{M13.13}$$

$$P^l_{ik} \geq B_i(\theta_{L^+_i k} - \theta_{L^-_i k}) + M(\neg z_{ik}) \qquad \forall i \in L \tag{M13.14}$$

$$P^l_{ik} \leq B_i(\theta_{L^+_i k} - \theta_{L^-_i k}) - M(\neg z_{ik}) \qquad \forall i \in L \tag{M13.15}$$

3.8.2 The Restoration Ordering Problem

Once a minimal set of items to repair is obtained, the Restoration Ordering Problem (ROP) determines the best order in which to repair them. The ROP ignores the routing aspects and the time to move from one location to another, which would couple the routing and power flow aspects. Instead, it views the restoration as a sequence of discrete steps and chooses which item to restore at each step. This subproblem is similar to the network re-energizing problem studied in power system restoration research but this model considers only the steady-state behavior, since this is the appropriate level for the PRVRP. Model 13 depicts the ROP model for the linearized DC model. The ROP contains

essentially $|R|$ flow models similar to those from Model 11, where R denotes the set of selected items to repair. These flows are linked through the decision variables o_{rk} that specify whether item r is repaired at step k. Constraint (M13.3) ensures that at most one item is repaired at each step, constraint (M13.4) ensures that an item remains repaired in future time steps, and constraint (M13.5) ensures that an item is activated only if it has been repaired. Constraint (M13.2) computes the flow at each step and the objective (M13.1) minimizes the sum of the differences between the maximum flow and the flow at each step. Constraints (M13.6–M13.15) are as in in Model 11 above.

For instances with more than 30 steps, this model can be too difficult to solve for state-of-the-art MIP solvers. Instead, we use a technique called Large Neighborhood Search (LNS) to find a near-optimal solution quickly (e.g., [92, 14]). The key idea underlying LNS is to fix parts of a solution in a structured but randomized way and to reoptimize over the remaining decision variables. This process is iterated until the solution has not been improved for some number of iterations. In the case of the ROP, LNS relaxes a particular subsequence, fixing the remaining part of the ordering, and reoptimizes the relaxed sequence. The reoptimization can use any optimization technology.

3.8.3 Vehicle Routing with Precedence Constraints

The ROP produces an ordering of the repairs that is used to inject precedence constraints on the jobs. This gives rise to a vehicle routing problem that implements a high-quality restoration plan while optimizing the dispatching itself. Note that the ROP is not used to impose a *total* ordering; instead, it merely injects a partial order among the jobs. Indeed, several repairs are often necessary to restore parts of the unserved demand; so that imposing a total order between these repairs reduces the flexibility of the routing and thus may degrade solution quality. As a result, the ROP solution partitions the set of repairs into a sequence of groups and the precedence constraints are imposed among the groups. The resulting Pickup and Delivery Vehicle Routing Problem with Precedence Constraints (PDVRPPC) consists in assigning a sequence of jobs to each vehicle, that satisfies the vehicle capacity and pickup and delivery constraints specified earlier, as well as the precedence constraints injected by the ROP. A precedence constraint $i \rightarrow j$ between job i and j is satisfied if $EDT_i \leq EDT_j$. The objective consists in minimizing the average repair time $\sum_{j \in J} EDT_j$. This objective approximates the true power restoration objective and is tight when all restoration actions restore similar amounts of power. When combined with constraint injection, this approximation works well in practice.

The constraint-programming formulation for the PDVRPPC problem presented in Model 14 is almost a direct translation of the problem specifications. The model is defined in terms of locations, i.e., the pickups, the deliveries, and the starting and ending locations of the vehicles. The decision variables associate with every location l the next location in the visit order, the vehicle visiting l, the load of the vehicle when it arrives at l, and the earliest delivery time for l. The successor variables make up a large circuit by connecting the vehicles together. The objective function minimizes the summation of the delivery times. Constraint (M14.2) eliminates subtours. Constraints (M14.3–7) initialize specific vehicles: their initial load and their delivery times are set to zero and their

Model 14 A Constraint-Programming Model for the PDVRPPC.

Let:

$W^+ = \{1 \dots d\}$	– pickups
$W^- = \{d+1 \dots 2d\}$	– dropoffs
$J = W^- \cup W^+$	– all restoration jobs
$H^+ = 2d+1 \dots 2d+m$	– vehicle departures
$H^- = 2d+m+1 \dots 2d+2m$	– vehicle returns
$L = W^- \cup W^+ \cup H^+ \cup H^-$	– all locations
$Pair : W^- \rightarrow W^+ s$	– the pickup associated with a dropoff
PC	– the precedence constraints from the ROP

Variables:

$\sigma[L] \in L$	– successor of a location
$vehicle[L] \in V$	– vehicle assignment of a location
$load[L] \in \{0, \dots, c\}$	– vehicle load at a location
$EDT[L] \in \{0, \dots, \infty\}$	– delivery time of a location

Minimize:

$$\sum_{i \in W^-} EDT[i] \qquad \text{(M14.1)}$$

Subject To:

$$circuit(\sigma) \qquad \text{(M14.2)}$$

for $l \in H^+$

$$vehicle[l] = vehicle[\sigma[l]] \qquad \text{(M14.3)}$$
$$load[l] = 0 \qquad \text{(M14.4)}$$
$$load[\sigma[l]] = 0 \qquad \text{(M14.5)}$$
$$EDT[l] = 0 \qquad \text{(M14.6)}$$
$$EDT[\sigma[l]] \geq T(l, \sigma[l]) + s(\sigma[l]) \qquad \text{(M14.7)}$$

for $l \in J$

$$vehicle[l] = vehicle[\sigma[l]] \qquad \text{(M14.8)}$$
$$load[\sigma[l]] = load[l] + d(l) \qquad \text{(M14.9)}$$
$$EDT[\sigma[l]] \geq s(\sigma[l]) + T(l, \sigma[l]) + EDT[l] \qquad \text{(M14.10)}$$

for $l \in W^-$

$$vehicle[Pair(l)] = vehicle[l] \qquad \text{(M14.11)}$$
$$EDT[Pair(l)] \leq EDT[l] \qquad \text{(M14.12)}$$

for $\langle i \rightarrow j \rangle \in PC$

$$EDT[i] \leq EDT[j] \qquad \text{(M14.13)}$$

first visit has the right vehicle, load, and delivery time. Constraints (M14.8–M14.10) specify the constraints for successors, which have the same vehicle, a modified load, and a larger delivery time. Constraints (M14.11) make sure that a (pickup,delivery) pair is served by the same vehicle and constraints (M14.12) ensure that pickups are performed before the delivery. Constraints (M14.13) inject the precedence from the ROP stage. In practice, the PDVRPPC model is solved using LNS and constraint programming. LNS and constraint programming are very effective for complex vehicle routing problems (e.g., [14, 15]). In contrast, traditional MIP systems have difficulty with the objective function of the PDVRPPC (e.g., [23]).

3.8.4 The Precedence Relaxation Problem

The last step of the algorithm is a post-processing optimization that relaxes some of the injecting constraints. Indeed, it may happen that vehicles end up waiting at some repair locations due to the precedence constraints. In such circumstances, it is almost always beneficial to relax the precedence constraints and let the vehicle perform its repairs earlier. This step assumes that the routes are fixed and determines which injected constraints should be relaxed to reduce the overall size of the blackouts. The model can be specified in terms of a linearized DC model very much like the ROP (e.g., by introducing variables specifying the time at which a repair is performed). Once again, the MIP model does not scale to large-scale disasters and our implementation also uses LNS.

3.9 Experimental Results

3.9.1 Benchmarks

The disaster scenarios are based on the U.S. infrastructure and were generated at Los Alamos National Laboratory using state-of-the-art hurricane simulation tools similar to those used by the National Hurricane Center [4]. The benchmarks are based on four different geographic locations. For a given location, the benchmarks share the same power and transportation infrastructure but differ in the damage scenarios. Each scenario is characterized by its damage to the power system and transportation infrastructures and is generated by the disaster simulation tools (e.g., weather and fragility simulations). This produces a total of 51 different benchmarks. Each geographic location has a power network containing about 300 items and about 13 repair crews are available for restoration. The number of damaged items ranges from 0 to 121. For simplicity, we group the benchmarks in three categories: small ($|J| < 20$), medium ($|J| < 50$), and large ($|J| \geq 50$). In total, there are 28 small, 14 medium, and nine large benchmarks. The large benchmarks are considerably more difficult than prior work in related areas. For example, the standard IEEE-118 benchmark has not been solved optimally in the context of optimal line switching [44] or network interdiction [90] (our MIP results for this difficulty level are consistent).

3.9.2 The Baseline and Relaxed Algorithms

To validate our results, we compare our PRVRP algorithm to a baseline algorithm modeling the practice in field that proceeds roughly as follows: (1) power engineers use their knowledge of the network to decide which key items to repair; (2) crews are dispatched to make the necessary repairs; (3) crews prefer to fix all broken items near the area to which they are dispatched. This process can be captured as an instance of our constraint-injection algorithm with the following choices: (1) the MRSP and ROP are solved with a greedy heuristic that incrementally chooses to repair the item bringing the largest increase in power flow; (2) the routing problem is identical to the PDVRPPC, except that the objective function seeks to minimize the total travel distance, not the sum of earliest delivery times. This captures the fact that each vehicle crew works independently to perform its

repairs as quickly as possible. Additionally, we calculate a relaxation of the PRVRP that assumes an infinite number of repair crews. In this relaxation, every restoration requires only the time for pickup, delivery, and repair. This relaxation provides an upper bound on the distance between our solution and the optimal solution.

The optimization algorithms were implemented in the COMET system [43, 73, 100] and the experiments were run on Intel Xeon CPU 2.80 GHz machines running 64-bit Linux Debian. The experiments use the standard linearized DC power flow equations for the power system modeling. Both a MIP- and an LNS-based formulation of the PRVRP problem are considered. The LNS approach is necessary for scaling to large instances. Due to fast-response requirements in disaster recovery, each subproblem is solved with a fixed time limit, so that a solution can be found in less than one hour. The time limits are as follows: two minutes for MSRP, 20 minutes for ROP, and 30 minutes for PDVRPPC. All algorithms require an LNS component to solve the routing aspect of the problem and hence the solutions may vary among runs. As a result, we report the mean value of five runs of each algorithm. Note also that, on four medium and four large benchmarks, the MIP solver cannot find a feasible solution to the ROP within the time limit.

3.9.3 Quality of the Results

Figures 3.7 and 3.8 present the final restoration results for one run of the algorithms on one medium and one large instance respectively. The MIP model is omitted from Figure 3.7 because a feasible solution to the ROP was not found within the time limit. The results show that the constraint-injection algorithm produces dramatic improvements over the practice in the field. Moreover, the results are often close to the infinite-vehicle relaxation, indicating that our algorithm finds near-optimal solutions. Space constraints prevent us from presenting similar results for all the benchmarks. Instead, we present an aggregation of these results for each benchmark size. The reported values are summed across all instances within a benchmark category and then scaled relative to the baseline algorithm. Table 3.7 presents the results for the PRVRP objective. The first three rows include the benchmarks that can be solved by all algorithms, while the last three rows include the benchmarks that could not be solved by the MIP-based constraint injection. The constraint-injection approaches consistently reduces the blackout area by 50% or more. Finally, the quality of the LNS-based constraint injection is reduced to a 30% improvement on the largest instances (i.e., the four that cannot be solved by the MIP-based constraint injection). This is caused by one specific benchmark with an atypical structure. Based on the current benchmarks, this peculiar structure does not appear to characterize PDVRPPC instances in general. Finally, because these instances are often two to three times larger than the medium-sized instances, additional time is required for the PDVRPPC stage of the algorithm. However, this is not a fundamental limitation of the approach. Recent work has demonstrated effective techniques for solving very large PDVRPPC instances [93].

Table 3.8 presents the quality results for the MRSP and ROP subproblems indicating that the LNS-based and MIP-based algorithms produce 10% improvements on the MRSP and between 40% and 60% improvements on the ROP. Moreover, the results indicate that using an LNS algorithm

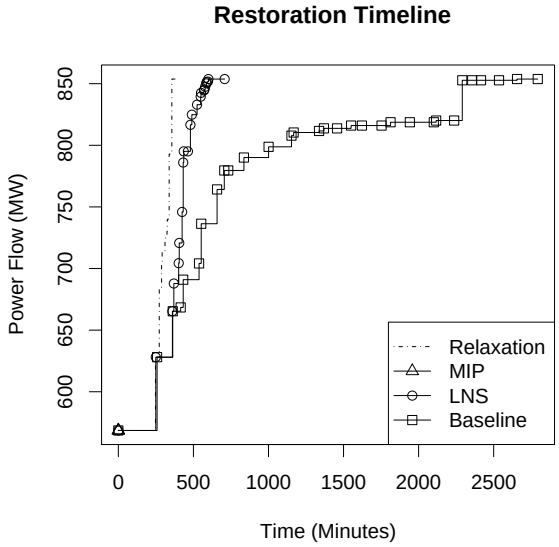

Figure 3.7: PRVRP Results Comparison (41 Damaged Items).

Restoration Objective					
Size (Count)		Baseline	MIP	LNS	Relaxation
Small	28	100%	46.1%	46.2%	34.6%
Medium	10	100%	31.3%	30.6%	21.4%
Large	5	100%	40.9%	46.8%	21.0%
Small	0	–	–	–	–
Medium	4	100%	–	47.3%	33.0%
Large	4	100%	–	69.7%	24.3%

Table 3.7: PRVRP Routing Quality.

not significantly degrade the quality of the MRSP and ROP solutions.

3.10 Power System Restoration Conclusion

This chapter has studied two novel problems in power system restoration, the Power System Stochastic Storage Problem (PSSSP) and Power Restoration Vehicle Routing Problem (PRVRP). The PSSSP captures the challenges of stockpiling components in order to recover from blackouts as well as possible after a disaster. PSSSPs are complex stochastic optimization problems combining power flow simulators, discrete storage decisions, discrete repair decisions given the storage decisions, and a collection of scenarios describing the potential effects of the disaster. This chapter proposed an exact mixed-integer formulation and a configuration-generation approaches for solving PSSSPs.

The PRVRP models the challenges of coordinating repair crews effectively so as to recover from blackouts as quickly as possible after a disaster. PRVRPs are complex, since they combine vehicle

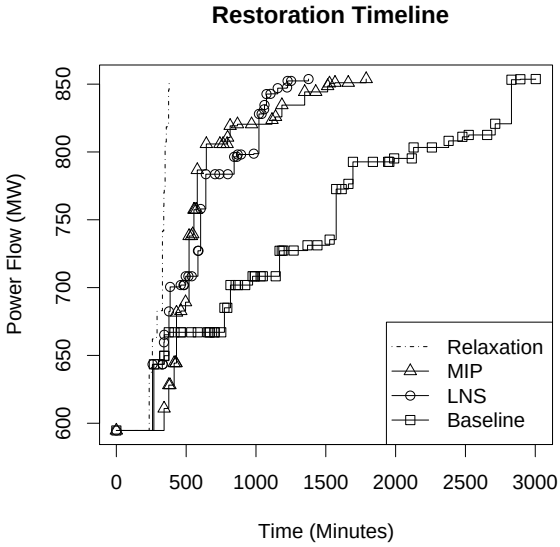

Figure 3.8: PRVRP Results Comparison (67 Damaged Items).

Size (Count)		Baseline	MIP	LNS
Average Restoration Set Size				
Small	28	7.64	6.79	7.04
Medium	10	25.3	23.2	23.9
Large	5	49	44.8	45.4
Restoration Order Quality				
Small	28	100%	59.3%	58.1%
Medium	10	100%	38.5%	38.7%
Large	5	100%	41.6%	52.3%

Table 3.8: PRVRP Subproblem Quality.

routing and power restoration scheduling problems. This chapter proposed a multi-stage optimization algorithm based on the idea of *constraint injection* that meets the aggressive runtime constraints necessary for disaster recovery. Together, the solution techniques for the PSSSP and PRVRP presented here implement the general solution framework from Chapter 1.

All of the proposed algorithms were evaluated on benchmarks produced by the Los Alamos National Laboratory, using the electrical power infrastructure of the United States. The disaster scenarios were generated by state-of-the-art hurricane simulation tools similar to those used by the National Hurricane Center.

The experimental results from Section 3.6 show that the configuration-generation algorithm for the PSSSP produces near-optimal solutions and produces orders of magnitude speedups over the exact formulation for large benchmarks. Moreover, both the exact and the configuration-generation formulations produce significant improvements over greedy approaches and hence should yield significant benefits in practice. The results also seem to indicate that the configuration-generation

algorithm should scale nicely to even larger disasters, given the small number of configurations necessary to reach a near-optimal solution. As a result, the configuration-generation algorithm should provide a practical tool for decision makers in the strategic planning phase just before a disaster strikes.

The experimental results from Section 3.9 demonstrate that the constraint-injection based algorithms for the PRVRP can reduce the blackouts by 50% or more over the practice in the field. Moreover, the results show that the constraint injection algorithm using large neighborhood search provides competitive quality and scales better than using a MIP solver on the subproblems. Together, the experiments validate quality and practicality of the proposed solution methods to the PSSSP and PRVRP and indicate that they can bring significant improvements to current field practices in power system restoration.

Chapter 4

Restoration of the Natural Gas and Power Networks

This chapter aims to solve the disaster restoration problem in the context of multiple interdependent infrastructures. It uses mixed-integer programs (MIP) for modeling interdependent power and gas networks, combining the linearized DC model (LDC) for the power network and a flow model for the gas network. The models are then integrated into the general restoration framework (Figure 1.2). The infrastructure interdependencies induce computational difficulties for MIP solvers in the prioritization step, which we address by using a randomized adaptive decomposition (RAD) approach. The RAD approach iteratively improves a restoration order by selecting smaller restoration subproblems that are solved independently. The proposed approach was evaluated systematically on a large collection of benchmarks generated with state-of-the-art hazard and fragility simulation tools on the U.S. infrastructure. The results demonstrate the scalability of the approach, which finds very high-quality solutions to large last-mile restoration problems and brings significant improvements over current field practices.

The rest of the chapter describes the modeling of multiple interdependent infrastructures and the approach for last-mile restoration of such infrastructures. It concludes by presenting experimental results.

4.1 Related Work

Disaster management and, in particular, the restoration of interdependent infrastructures are inherently interdisciplinary as they span the fields of reliability engineering, vulnerability management, artificial intelligence, and modeling of complex systems. The importance of interdependent infrastructure restoration was recognized soon after the 2001 World Trade Center attack [105] and this recognition has continued to spread over the past decade [28, 41, 22]. Interdependence studies in the reliability engineering area [41, 79] have focused primarily on topological properties such as

betweenness and *connectivity loss*. In artificial intelligence, to the best of our knowledge, restoration of interdependent infrastructures has not been studied, although power system restoration has been considered [18, 54, 13]. Although these methods are an excellent application of planning, configuration, and diagnosis techniques, they also use *connectivity* as the primary power model. These topological metrics provide some sufficient conditions for infrastructure operations. However, their fidelity is insufficient to incorporate line capacity constraints that are critical in modeling the pipeline compressor interdependencies studied here. Furthermore, the accuracy of topological metrics for models of infrastructure systems has recently been questioned [59] and the benefits of flow-based models of infrastructure systems are increasingly recognized in the reliability engineering community. [40].

References [69, 70, 49, 25] are the closest related work and warrant a detailed review. Our algorithms differ fundamentally from these earlier studies because of their use of the more accurate LDC for power systems, their scalability, and their application to cyclic interdependencies. Reference [69] is a good background paper on the nature and classification of various interdependencies. Early work focused on solving the MRSP [70] and considered the NYC's power, telephone, and subway infrastructure but was concerned only with restoring the power infrastructure. [49] assumed that restoration tasks have predefined due dates and developed a logic-based Benders decomposition for a weighted sum of different competing objectives. The impact on the actual infrastructure is not taken into account. [25] tried to solve jointly the multi-machine model of [49] and the MRSP [70], studying only the restoration of the power grid. Computation times are between three and 18 hours, with optimality gaps of 0.4% and 3.0% respectively. They report that using the MRSP instead of the full damage decreases the quality of the solution by 4.5%. This is about twice as bad as the worst-case effect in the formulation presented here. It is worth noting that, in damage scenarios for which the optimal solution is known, this approach's MRSP/ROP decoupling rarely cuts off the optimal solution.

4.2 Infrastructure Modeling

Power and gas infrastructures can be modeled and optimized at various levels of abstraction. Linear approximations are typically used for applications involving topological changes, a design choice followed here as well. This section presents a demand maximization model for interdependent power and gas infrastructures, which is a key building block in the restoration models.

The Power Infrastructure The power infrastructure is modeled in terms of the Linearized DC Model (LDC), a standard tool in power systems (e.g., [109, 68, 107, 82, 48]). In the LDC, a *power network* $\mathcal{PN}$ is represented by a collection of buses B and a collection of lines L connecting the buses. Each bus $i \in B$ may contain multiple generation units B_i^g and multiple loads B_i^o, and $B^g = \bigcup_{i \in B} B_i^g$ and $B^o = \bigcup_{i \in B} B_i^o$ denote the generators and loads across all buses. Each generator $j \in B^g$ has a maximum generation value $\hat{E}_j^g$ and each load $k \in B^o$ has a maximum consumption value $\hat{E}_k^o$. Each

Model 15 Power System Demand Maximization.

Inputs:

$\mathcal{PN} = \langle B, L, s \rangle$ – the power network

Variables:

$\theta_i \in (-\Re, \Re)$ – phase angle on bus i

$E_i^g \in (0, \hat{E}_i^g)$ – power injected by generator i

$E_i^o \in (0, \hat{E}_i^o)$ – power consumed by load i

$E_i^l \in (-\hat{E}_i^l, \hat{E}_i^l)$ – power flow on line i

Maximize:

$$\sum_{i \in B^o} E_i^o \tag{M15.1}$$

Subject to:

$$\theta_s = 0 \tag{M15.2}$$

$$\sum_{j \in B_i^o} E_j^o = \sum_{j \in B_i^g} E_j^g + \sum_{j \in LI_i} E_j^l - \sum_{j \in LO_i} E_j^l \quad \forall i \in B \tag{M15.3}$$

$$E_i^l = b_i(\theta_{L_i^+} - \theta_{L_i^-}) \quad \forall i \in L \tag{M15.4}$$

line $i \in L$ is assigned a *from* and *to* bus denoted by L_i^+ and L_i^- respectively and is characterized by two parameters: a maximum capacity $\hat{E}_i^l$ and a susceptance b_i. LO_j and LI_j respectively denote all the lines oriented *from* or *to* a given bus j. Lastly, one bus s is selected arbitrarily as the *slack bus* to remove numerical symmetries. Model 15 presents a LDC for maximizing the load of a power network $\mathcal{PN} = \langle B, L, s \rangle$. The decision variables are: (1) the phase angles of the buses θ; (2) the production level of each generator E^g; (3) the consumption level of each load E^o; (4) the flow on each line E^l, which can be negative to model a flow in the reverse direction. The objective (M15.1) maximizes the total load served. Constraint (M15.2) fixes the phase angle of the slack bus. Constraint (M15.3) ensures flow conservation (i.e., Kirchhoff's Current Law) at each bus, and constraint (M15.4) ensures the line flows are defined by line susceptances.

The Gas Infrastructure We use a network flow model for the gas system, also common in practice (e.g., [24, 74]). The gas model is similar to the power model. A *gas network* $\mathcal{GN}$ is represented by a collection of junctions J and a collection of pipelines P connecting the junctions. Each junction $i \in J$ may contain multiple generation units J_i^g (aka well fields) and multiple loads J_i^o (aka city gates) and we define J^g and J^o as in the power system. Each generator $j \in J^g$ has a maximum generation value $\hat{G}_j^g$ and each load $k \in J^o$ has a maximum consumption value $\hat{G}_k^o$. Each pipeline $i \in P$ is associated with a *from* and *to* junction, denoted by P_i^+ and P_i^- respectively, and a flow limit of $\hat{G}_i^p$. The sets PO_j and PI_j are defined as in the power system. Gas networks also have compressors that are denoted by the set PC. It is convenient to refer to the set of all pipelines attached to a compressor $i \in PC$ as P_i^c. The effects of compressors is significant only for the interdependent model and are a perfect example of why modeling interdependencies are so critical in finding high-quality restoration plans. Model 16 presents a linear program for maximizing the demand in a gas network. The inputs are a gas network $\mathcal{GN} = \langle J, P \rangle$ and the decision variables are: (1) the production level of each generator G^g; (2) the consumption level of each load G^o; (3)

Model 16 Gas System Demand Maximization.

Inputs:

$\mathcal{GN} = \langle J, P \rangle$ – the gas network

Variables:

$G_i^g \in (0, \hat{G}_i^g)$ – gas injected by well field i

$G_i^o \in (0, \hat{G}_i^o)$ – gas consumed by city gate i

$G_i^p \in (-\hat{G}_i^p, \hat{G}_i^p)$ – gas flow on pipeline p

Maximize:

$$\sum_{i \in J^o} G_i^o \qquad \text{(M16.1)}$$

Subject to:

$$\sum_{j \in J_i^o} G_j^o = \sum_{j \in J_i^g} G_j^g + \sum_{j \in PI_i} G_j^p - \sum_{j \in PO_i} G_j^p \quad \forall i \in J \qquad \text{(M16.2)}$$

the flow on each pipeline G^p (which can be negative as well). The objective (M16.1) maximizes the total loads served. Constraint (M16.2) ensures flow conservation at each junction. Independently, both models are linear programs (LP).

The Interdependent Power and Gas Infrastructure The power and gas networks have different types of interdependencies. *Sink-source* connections are common. For example, a gas city gate G^o can fuel a gas turbine engine that is an electric generator E^g. *Sink-sink* connections also appear: for example, a city gate G^o requires some energy from a load E^o to regulate its valves. All these interdependencies can be modeled in terms of implications $a \rightarrow c$ indicating that consequent c is not operational whenever antecedent a is not served at full capacity. *Pipeline compressors* also induce fundamental interdependencies. Indeed, compressors consume electricity from a load E^o to increase the pressure on a pipeline P, since sufficient line pressure is a feasibility requirement for the gas network. This dependency is modeled as a capacity reduction, since pressure is not captured explicitly in the linear gas model.

An interdependent model is inherently multi-objective. In practice, however, policy makers typically think of infrastructure restoration in terms of financial or energy losses. Both cases are naturally modeled as a linear combination of the power and gas objectives. The objectives consider only the set of loads in the networks that are not antecedent to a dependency. We use $T^e \subseteq B^o$ and $T^g \subseteq J^o$ to denote the filtered loads for the power and gas networks. If W^e and W^g are the weights of the infrastructures, then the joint objective is $W^e \sum_{i \in T^e} E_i^o + W^g \sum_{i \in T^g} G_i^o$. The maximal demand satisfaction of each network is often useful, and we use M^e and M^g to refer to the maximum power and gas demand satisfaction respectively.

We are almost in a position to present the interdependent model. The missing piece of information is the recognition that, whenever a component is not active, it may induce other components to be non-operational as well. For example, if a bus is inactive, then the components connected to that bus (e.g., lines, generators, loads) are non-operational too. These intra-network dependencies, which are modeled in terms of logical constraints, are not present in the demand maximization model for

Model 17 Interdependent Demand Maximization.

Inputs:

$\mathcal{PN} = \langle B, L, s \rangle$ – the power network
$\mathcal{GN} = \langle J, P \rangle$ – the gas network
A, A^c, C – the interdependencies
T^e, T^g – the demand points
W^e, W^g – the demand weights

Variables:

$y_i \in \{0, 1\}$ – item i is activated
$z_i \in \{0, 1\}$ – item i is operational
$fl_i \in \{0, 1\}$ – all of item i's load is satisfied
$\theta_i \in (-\Re, \Re)$ – phase angle on bus i
$E^g_i \in (0, \hat{E}^g_i)$ – power injected by generator i
$E^o_i \in (0, \hat{E}^o_i)$ – power consumed by load i
$E^l_i \in (-\hat{E}^l_i, \hat{E}^l_i)$ – power flow on line i
$G^g_i \in (0, \hat{G}^g_i)$ – gas injected by well field i
$G^o_i \in (0, \hat{G}^o_i)$ – gas consumed by city gate i
$G^p_i \in (-\hat{G}^p_i, \hat{G}^p_i)$ – gas flow on pipeline p

Maximize:

$$W^e \sum_{d \in T^e} E^o_d + W^g \sum_{d \in T^g} G^o_d \qquad \text{(M17.1)}$$

Subject to:

$$y_i = 1 \qquad \forall i \in N \setminus C \qquad \text{(M17.2.1)}$$
$$fl_i \Leftrightarrow \hat{I}^o_i \leq I^o_i \qquad \forall i \in A \qquad \text{(M17.2.2)}$$
$$y_i = \bigwedge_{j \in A^c_i} fl_j \qquad \forall i \in C \qquad \text{(M17.2.3)}$$

$$z_i = y_i \qquad \forall i \in B \qquad \text{(M17.3.1)}$$
$$z_i = y_i \wedge y_j \qquad \forall j \in B\ \forall i \in B^g_j \cup B^o_j \qquad \text{(M17.3.2)}$$
$$z_i = y_i \wedge y_{L^+_i} \wedge y_{L^-_i} \qquad \forall i \in L \qquad \text{(M17.3.3)}$$
$$\theta_s = 0 \qquad \text{(M17.3.4)}$$
$$\sum_{j \in B^o_i} E^o_j = \sum_{j \in B^g_i} E^g_j + \sum_{j \in LI_i} E^l_j - \sum_{j \in LO_i} E^l_j \qquad \forall i \in B \qquad \text{(M17.3.5)}$$
$$\neg z_i \rightarrow E^g_i = 0 \qquad \forall i \in B^g \qquad \text{(M17.3.6)}$$
$$\neg z_i \rightarrow E^o_i = 0 \qquad \forall i \in B^o \qquad \text{(M17.3.7)}$$
$$\neg z_i \rightarrow E^l_i = 0 \qquad \forall i \in L \qquad \text{(M17.3.8)}$$
$$z_i \rightarrow E^l_i = B_i(\theta_{L^+_i} - \theta_{L^-_i}) \qquad \forall i \in L \qquad \text{(M17.3.9)}$$

$$z_i = y_i \qquad \forall i \in J \cup PC \qquad \text{(M17.4.1)}$$
$$z_i = y_i \wedge y_j \qquad \forall j \in J\ \forall i \in J^g_j \cup J^o_j \qquad \text{(M17.4.2)}$$
$$z_i = y_i \wedge y_{P^+_i} \wedge y_{P^-_i} \qquad \forall i \in P \qquad \text{(M17.4.3)}$$
$$\sum_{j \in J^o_i} G^o_j = \sum_{j \in J^g_i} G^g_j + \sum_{j \in PI_i} G^p_j - \sum_{j \in PO_i} G^p_j \qquad \forall i \in J \qquad \text{(M17.4.4)}$$
$$\neg z_i \rightarrow G^g_i = 0 \qquad \forall i \in J^g \qquad \text{(M17.4.5)}$$
$$\neg z_i \rightarrow G^o_i = 0 \qquad \forall i \in J^o \qquad \text{(M17.4.6)}$$
$$\neg z_i \rightarrow G^p_i = 0 \qquad \forall i \in P \qquad \text{(M17.4.7)}$$
$$\neg z_i \rightarrow -\tilde{G}^p_j \leq G^p_j \leq \tilde{G}^p_j \qquad \forall j \in P^c_i\ \ \forall i \in PC \qquad \text{(M17.4.8)}$$

a single infrastructure. Computationally, they imply that demand maximization of interdependent infrastructures become a MIP model, instead of a LP. The complete demand maximization model for the interdependent power and gas infrastructure is presented in Model 17. For clarity, we use the logical constraints, not their linearizations (which can be obtained through standard transformations). The inputs are specified in terms of following additional notation: N is the collection of all of the infrastructure components, i.e., $N = B \cup B^g \cup B^o \cup L \cup J \cup J^g \cup J^o \cup P \cup PC$; the sink-sink and sink-source interdependencies are specified by antecedent and consequent relations. The set A is the collection of all antecedent items and C is the set of all consequent items; for each consequent $i \in C$ the set $A_i^c \subseteq A$ denotes all antecedents of i. The collection of all load points in both infrastructures is $I^o = G^o \cup E^o$, and $\hat{I}_i^o$ is the maximum load of a resource $i \in I^o$. The model inputs are then given by the network $\mathcal{IN} = \langle \mathcal{PN}, \mathcal{GN}, A, A^c, C, T^e, T^g, W^e, W^g \rangle$. The variables include those described in Models 15 and 16 and the objective function (M17.1) in Model 17 was described earlier.

To model the effect of the interdependencies on the network topologies, a binary variable y_i associated with each component $i \in N$ denotes whether the component is active. Another variable z_i is associated with component i to denote whether component i is operational. Most of the y_i variables are set to one: only those affected by interdependencies may be zero, as per Constraint (M17.2.1). The antecedent i of a dependency is always a load point and is operational only when its load is at full capacity; this is captured by binary variable fl_i and Constraint (M17.2.2): that is, $fl_i = 1$ if and only if $\hat{I}_i^o \leq I_i^o$. Constraint (M17.2.3) specifies that each consequent $i \in C$ is active if all of its antecedents A_i^c are at full capacity. Constraint (M17.4.8) specifies the capacity reduction of a compressor-dependent pipeline $j \in P_i^c$ when its compressor $i \in PC$ is not operational, i.e., $z_i = 0$. Note that the regular operating capacity of pipeline j is $\hat{G}_j^p$, while its reduced capacity is $\tilde{G}_j^p$.

Constraints (M17.3.1–M17.3.9) model the power system. Constraints (M17.3.1–M17.3.3) describe which components are operational according to the operational rules sketched out previously. Constraints (M17.3.4) and (M17.3.5) are from Model 15. Constraints (M17.3.6–M17.3.9) impose restrictions on power flow, consumption, and production depending on the operational state: They ensure that a non-operational generator, load, or line cannot produce, consume, or transmit power. Constraints (M17.4.1–M17.4.8) model the gas system. The principles are the same as in the power system except for constraints (M17.4.8), which model the effects of non-operational compressors as discussed previously.

4.3 Joint Infrastructure Repair and Restoration

The joint repair and restoration of an interdependent infrastructure is extremely challenging computationally. It is a multiple pickup and delivery vehicle routing problem, whose objective function is defined in terms of a series of demand maximization problems, one for each repair action. Each of these demand maximizations is a MIP, which leads to an overall intractable formulation. Indeed, for even a single infrastructure, where demand maximization is a LP, tackling the problem globally is beyond the scope of existing MIP solvers. For this reason, we follow the multi-stage approach

MULTI-STAGE-IRVRP(*Network* $\mathcal{IN}$, *IRVRP* G)
1 $\mathcal{R} \leftarrow MinimumRestorationSetProblem(\mathrm{G}, \mathcal{IN})$
2 $\mathcal{O} \leftarrow RestorationOrderProblem(\mathcal{IN}, \mathcal{R})$
3 **return** $PrecedenceRoutingProblem(\mathrm{G}, \mathcal{O})$

Figure 4.1: The Multi-Stage IRVRP Algorithm.

proposed in [104], which produces high-quality solutions to the joint repair and restoration of the power system, even for large instances.

The multi-stage approach has thef three steps depicted in Figure 4.1. As inputs, the Infrastructure Restoration Vehicle Routing Problem (IRVRP) requires an infrastructure network $\mathcal{IN}$ and an IRVRP instance G, which contains the network damage information and other data necessary for constructing the vehicle routing problem. The first step is a minimum restoration set problem that determines a smallest set of items to restore the infrastructure to full capacity. The second set is a restoration order problem that produces the order in which the components must be repaired. This order produces precedence constraints that are injected into the pickup and delivery routing problem to produce the restoration plan. Only the first two steps are affected when an interdependent power and gas infrastructure is considered and here we study only these two steps.

4.4 The Minimum Restoration Set Problem

The Minimum Restoration Set Problem (MRSP) determines a smallest set of items needed to restore the network to full capacity (Model 18). The optimization builds extensively on Model 17 but has four significant changes. First, additional inputs are necessary, namely the set of damaged components $D \subseteq N$. Second, the objective (M18.1) now minimizes the number of repairs. Third, constraints (M18.2 and M18.3) ensure that the network operates at full capacity. Fourth, constraint (M18.4) ensures that only undamaged items are activated. The remaining constraints are identical to (M17.2.2–M17.4.8) in Model 17.

4.5 The Restoration Ordering Problem

Once a set $R \subseteq D$ of items to repair is obtained, the Restoration Ordering Problem (ROP) determines the best order in which to repair the items. The ROP ignores the routing aspects and the duration to move from one location to another, which would couple the routing and demand maximization aspects. Instead, it views the restoration as a sequence of discrete steps and chooses which item to restore at each step. Model 19 depicts the ROP model for interdependent infrastructures. The ROP essentially duplicates Model 17 $|R|$ times, where R is the set of selected items to repair. These models are linked through the decision variables y_{ki} specifing whether item i is repaired at step k. Constraint (M19.2) ensures that undamaged items are activated, constraint (M19.3) makes sure that at most one item is repaired at each step, and constraint (M19.4) ensures that an item remains repaired in subsequent steps. The objective (M19.1) maximizes the satisfied demands at each step.

Model 18 MRSP for Interdependent Infrastructures.

Inputs:

M^e, M^g – the maximum demands in undamaged networks
D – the set of damaged items
All inputs from Model 17

Variables:

Identical to Model 17

Minimize:

$$\sum_{i \in D} y_i \qquad \text{(M18.1)}$$

Subject to:

$$\sum_{d \in T^e} E^o_d \geq M^e \qquad \text{(M18.2)}$$

$$\sum_{d \in T^g} G^o_d \geq M^g \qquad \text{(M18.3)}$$

$$y_i = 1 \quad \forall i \in N \setminus (C \cup D) \qquad \text{(M18.4)}$$

Constraints (M17.2.2–M17.4.8) from Model 17

Model 19 ROP Model for Interdependent Infrastructures.

Inputs:

R – the set of items to restore
D – the set of damaged items
All inputs from Model 17

Variables:

Variables of Model 17 replicated $|R|$ times

Maximize:

$$\sum_{k=1}^{|R|} W^e \sum_{d \in T^e} E^o_{kd} + W^g \sum_{d \in T^g} G^o_{kd} \qquad \text{(M19.1)}$$

Subject to: $(1 \leq k \leq |R|)$

$$y_{ki} = 1 \qquad \forall i \in N \setminus (C \cup D) \qquad \text{(M19.2)}$$

$$\sum_{i \in R} y_{ki} = k \qquad \text{(M19.3)}$$

$$y_{(k-1)i} \leq y_{ki} \quad \forall i \in R \qquad \text{(M19.4)}$$

k replicates of constraints (M17.2.2-M17.4.8) from Model 17

The remaining model constraints are identical to (M17.2.2–M17.4.8) in Model 17 but are replicated for each of the k models.

The ROP model is significantly more challenging for interdependent infrastructures because the demand maximization problem is now a MIP instead of an LP, as is the case for a single infrastructure. MIP solvers have significant scalability issues, mainly because the ROP generalizes the transmission switching problem, known to be extremely challenging for state-of-the-art MIP solvers (e.g., [44]).

4.6 Randomized Adaptive Decouplings

To overcome these computational difficulties, we use a Randomized Adaptive Decoupling (RAD) scheme. RAD schemes have been found useful in a variety of applications in logistics [16], scheduling [81], and disaster management [93].

Informal Presentation First observe that the ROP can be viewed as a function $ROP : R \times D \rightarrow \mathcal{O}$ that, given a set R of components to repair and a set of damaged components D ($R \subseteq D$), produces an ordering $\mathcal{O}$ of R maximizing the satisfied demands over time. The RAD scheme repeats the following two steps:

1. Partition the sequence $\mathcal{O}$ into the subsequences $S_1, \ldots, S_l$, i.e., $\mathcal{O} = S_1 :: S_2 :: \ldots :: S_l$, where :: denotes sequence concatenation.

2. Solve an ROP problem, called the Priority Restoration Order Problem (PROP), in which the items in S_j must be scheduled before the items in S_{j+1} ($1 \leq j < l$).

Obviously, the PROP produces a lower bound to the ROP. However, it enjoys a nice computational property: it can be solved by solving a sequence of smaller decoupled ROPs defined as

$$
\begin{aligned}
&ROP(S_1, D) \\
&\ldots \\
&ROP(S_i, D \setminus (S_1 \cup \ldots \cup S_{i-1})) \\
&\ldots \\
&ROP(S_l, D \setminus (S_1 \cup \ldots \cup S_{l-1})).
\end{aligned}
$$

The RAD scheme then starts from a solution $\mathcal{O}_0$ obtained by a standard utilization heuristic. At iteration i, the scheme has a solution $\mathcal{O}_i$ that is partitioned to obtain a PROP $\mathcal{P}_i$; $\mathcal{P}_i$ is solved by exploiting the decoupling to obtain a solution $\mathcal{O}_{i+1}$. The successive solutions satisfy

$$\mathcal{O}_0 \leq \mathcal{O}_1 \leq \ldots \leq \mathcal{O}_i \leq \ldots$$

The RAD scheme also ensures that the random partition of a solution σ into $S_1 :: S_2 :: \ldots :: S_l$ produces subsequences of length between two parameters s and S in order to generate ROPs that are non-trivial and computationally tractable. May options are available for the *stoppingCriteria()* function. We found that a combination of a fixed time limit and 10 iterations without improvement saved time on easier problems.

Formalization The RAD algorithm for the ROP is depicted in Figure 4.2. Observe that the partition uses the current solution $\mathcal{O}$ and that the PROP never degrades the solution quality since $\mathcal{O}$ is a solution to the PROP. The algorithm could be easily generalized to a variable neighborhood search [55] by increasing the sequence size, e.g.,

$$S = (1 + \alpha)S,$$

ROP-RAD$(R, D, [s, S])$
1 $\mathcal{O} \leftarrow$ *ROP-UTIL*(R, D);
2 **while** $\neg$*stoppingCriteria*()
3 **do** $\langle S_1, \ldots, S_l \rangle \leftarrow$ *RandomPartition*$(\mathcal{O}, [s, S])$;
4 $\quad \mathcal{O} \leftarrow$ *PROP*$(\langle S_1, \ldots, S_l \rangle, D)$;
5 **return** $\mathcal{O}$;

Figure 4.2: The RAD algorithm for the ROP.

when no improvement to the solution is found after several iterations. This was not necessary, however, to obtain high-quality solutions on our benchmarks. The PROP can be formally defined as follows.

Definition 1 (PROP)**.** *Given $S_1 \cup \ldots \cup S_l \subseteq D$, the Priority Restoration Order Problem $PROP(\langle S_1, \ldots, S_l \rangle, D)$ is a ROP problem $ROP(S_1 \cup \ldots \cup S_l, D)$ with the following additional constraints $(1 \leq j \leq l)$:*

$$\forall i \in S_j : y_{ti} = 1 \text{ where } t = \sum_{n=1}^{j} |S_n| \tag{4.1}$$

Observe that a consequence of these constraints is that all items in $S_1, \ldots, S_j$ are repaired before the items in S_{j+1} $(1 \leq j < l)$. We now show that the PROP can be decomposed into a set of independent ROPs.

Theorem 1. *A Priority Restoration Ordering Problem $\mathcal{P} = PROP(\langle S_1, \ldots, S_l \rangle, D)$ can be solved optimally by solving l independent ROPs:*

$$\begin{aligned}
&\mathcal{R}_1 = ROP(S_1, D) \\
&\ldots \\
&\mathcal{R}_i = ROP(S_i, D \setminus (S_1 \cup \ldots \cup S_{i-1})) \\
&\ldots \\
&\mathcal{R}_l = ROP(S_l, D \setminus (S_1 \cup \ldots \cup S_{l-1})).
\end{aligned}$$

Proof. It is sufficient to show that the union of the objectives and constraints of the l independent ROPs is equivalent to the original PROP, $\mathcal{P}$. The objective equivalence follows from the fact that the sum of the objective functions of $\mathcal{R}_1, \ldots, \mathcal{R}_l$ is the objective function of $\mathcal{P}$. The system of constraints is more interesting. The additional constraints of the PROP produce four properties. Consider a subsequence S_j and let $s = 1 + \sum_{n=1}^{j-1} |S_n|$ and $t = \sum_{n=1}^{j} |S_n|$. Then the following properties hold:

$$\begin{aligned}
y_{si} &= 1 \qquad \forall i \in (S_1 \ldots \cup S_{j-1}) \\
y_{si} &= 0 \qquad \forall i \in (S_j \ldots \cup S_l) \\
y_{ti} &= 1 \qquad \forall i \in (S_1 \ldots \cup S_j) \\
y_{ti} &= 0 \qquad \forall i \in (S_{j+1} \ldots \cup S_l).
\end{aligned}$$

These follow from induction on the PROP constraints (4.1) and (M19.4) and are enforced in the l

independent ROPs through careful selection of the restoration and damage sets:

$$\mathcal{R}_j = ROP(S_j, D \setminus (S_1 \cup \ldots \cup S_{j-1})).$$

Substitution in the ROP model of constraints (M19.2) yields

$$y_{ki} = 1 \;\; \forall i \in N \setminus (C \cup D \setminus (S_1 \cup \ldots \cup S_{j-1})),$$

which ensures all the y variables satisfy the first PROP property at time s. By definition, the ROP will restore only the items in the restoration set. Assigning the restoration set to S_j ensures that the remaining PROP properties hold.

Constraints (M19.4) in the PROP ensure that, once an item is repaired, it remains repaired. The key observation for this constraint is to look at the y_{ki} variables in s-to-t intervals, i.e., $[s_1..t_1][s_2..t_2]\ldots[s_l..t_l]$. For one of these intervals $[s_i..t_i]$, the ROPs enforce precedence constraints among

$$y_{(k-1)e} \leq y_{ke} \;\; \forall e \in S_i \; k \in [s_i + 1..t_i].$$

We now show that the remaining inequalities in the PROP can be removed when the four PROP properties are enforced. First, we know that all elements in S_i are repaired after time t_i, i.e.,

$$y_{ke} = 1 \;\; \forall e \in S_i \; , k \geq t_i.$$

Hence, all subsequent inequalities in this interval are guaranteed to be satisfied and can be removed. We also know that all elements in S_i were not repaired before time s_i, i.e.,

$$y_{ke} = 0 \;\; \forall e \in S_i \; , k < s_i.$$

Hence, all the previous inequalities in this interval are guaranteed to be satisfied and can be removed. Applying these simplifications for all intervals $1 \leq j \leq l$ reveals that the ROPs enforce all of the relevant constraints.

constraints (M19.3) in PROP ensures that at most one item is repaired at every time step. Using arithmetic transformations, the original constraint can be rewritten in terms of the subsequences

$$\sum_{i \in R} y_{ki} = k \; \forall k \in [1..|R|]$$

$$\sum_{j=1}^{l} \sum_{i \in S_j} y_{ki} = k \; \forall k \in [1..|R|].$$

By the PROP properties, for any subsequence S_j, we know all of the elements in $S_1, \ldots, S_{j-1}$ have been set to 1 and all of the elements in $S_{j+1}, \ldots, S_l$ have been set to 0. That is,

$$\sum_{m=1}^{j-1} \sum_{i \in S_m} y_{ki} = \sum_{m=1}^{j-1} |S_m|$$

$$\sum_{m=j+1}^{l} \sum_{i \in S_m} y_{ki} = 0.$$

Observe that $1..|R|$ can be partitioned into l s-to-t intervals $[s_1..t_1], [s_2..t_2], \ldots, [s_l..t_l]$. Given the previous formulas, the PROP constraints (M19.3) for subsequence S_j are

$$\sum_{m=1}^{j-1} |S_m| + \sum_{i \in S_j} y_{ki} + \sum_{m=j+1}^{l} 0 = k \; \forall k \in [s_j..t_j].$$

After expanding the definition of $[s_j..t_j]$, the constant term $\sum_{n=1}^{j-1} |S_n|$ may be removed by changing the interval range:

$$\sum_{i \in S_j} y_{ki} = k \; \forall k \in [1..|S_j|].$$

In this way, constraints (M19.3) becomes j disjoint constraints in the PROP model that are enforced in ROPs.

The optimal solution of $\mathcal{P}$ can thus be obtained by concatenating the optimal solution of the subproblems $\mathcal{R}_1, \ldots, \mathcal{R}_l$. □

The Utilization Heuristic The utilization heuristic used by the RAD algorithm (Figure 4.2) is designed to approximate current best practices for prioritizing repairs. In existing best practices, each infrastructure provider works independently and prioritizes repairs based on the percentage of network flow used by each element under normal operating conditions. This measure is called the *utilization* of the element. Because each utility works independently, each infrastructure system is solved independently using Models 15 and 16. Given a flow on the power network f_e or gas network f_g, the utilization of these components is f_e/M^e and f_g/M^g respectively. Each infrastructure provider prioritizes repairs based on the greatest utilization values. Given that the utilization value is unitless, these restoration priorities may be extended to the multi-infrastructure domain by using the weighting factors W^e and W^g. This greedy heuristic serves both as a seed for our hybrid optimization approach and as the basis for comparison. The experimental section below demonstrates that optimization brings significant improvements over this current best practice.

Computational Considerations The RAD approach should be contrasted with a local search approach that would swap items in the current ordering. Such a local search is computationally expensive, since a swap between items in positions i and j requires solving $(j - i + 1)$ Model 17 instances, which are MIP models. Moreover, the complexity of these MIP models makes it hard to determine which moves are attractive in the local search and thus forces the local search to examine a large number of costly moves. In contrast, the RAD scheme exploits temporal locality, the subsequences are small, and the MIP solver uses the linear relaxation to guide the large neighborhood exploration.

Practical Considerations In practice, even some ROP problems with fewer than 10 items can be challenging to solve optimally and may take several minutes. For this reason, our RAD scheme uses a time limit on the subproblems and does not always solve the ROPs optimally. It is also useful

to point out that, in practice, all the decoupled ROPs can be solved in parallel. This feature was not used in our implementation but would be highly beneficial in practice.

4.7 Experimental Results

The benchmarks were produced by Los Alamos National Laboratory and are based on U.S. power and gas infrastructures. The disaster scenarios were generated using state-of-the-art hurricane simulation tools used by the National Hurricane Center [4, 88]. The power network has 326 components and the gas network has 93 components. Network damages range from 10 to 120 components. The experiments were run on Intel Xeon 2.8 GHz processors on Debian Linux. The algorithms were implemented in the COMET system using SCIP as a MIP solver. The execution times were limited to one hour to be compatible with the disaster recovery context. The weighting parameters W^e, W^g were selected to balance the demands of the networks in percents ($W^e = 0.5/M^e, W^g = 0.5/M^g$), where M^e, M^g are the maximal demand satisfaction of the power and gas networks respectively (these results are consistent for other values of W^e and W^g). The subsequences in the decomposition are of sizes between 4 and 8.

Our approach is compared to the *utilization heuristic* that approximates the current best practices in multiple infrastructure restoration. The experiments focus only on the ROP problem, which is the bottleneck of the approach. As mentioned earlier, the final routing is not affected by considering multiple interdependent infrastructures. Tables 4.1 and 4.2 present the quality and run time data from the various ROP algorithms on 33 damage scenarios. The results are first grouped into ROP and MRSP+ROP to show the benefits of including the MRSP stage. Column $|D|$ is the number of damaged items, *MIP* is the restoration result using Model 19, *RAD* is the restoration result using the decomposition from Figure 4.2, and $|R|$ is the restoration set size after using the MRSP. The values in the MIP and RAD columns indicate the multiplicative improvement over the utilization heuristic. For example, a value of 2.0 indicates that the optimization algorithm doubled the amount of satisfied demands over the time of the restoration (i.e., reduced the size of the power and gas "blackout" by 2). An asterisk indicates a proof of optimality. Entries are omitted for the MIP when no solution was found within the time constraints. The aggregate statistics at the bottom of the Table 4.1 summarize the results. Due to the incomplete MIP data, several subsets are of interest: MIP-μ is the set of instances that the MIP can solve; MRSP MIP-μ is the set of instances that the MIP can solve when the MRSP is used; RAD-μ is the set of all instances; and Large RAD-μ is the set of instances where $|D| \geq 40$. To provide an intuition behind the numbers reported in Table 4.1, Figure 4.3 depictes the detailed restoration plans on Benchmark 20 for the utilization heuristic, the MIP approach, and the RAD algorithm. The figure shows the significant benefits provided by optimization technologies in general and the RAD approach in particular.

The results indicate that the RAD approach significantly improves the practice in the field and more than doubles the level of service within the time constraints. The instances without the MRSP stage are particularly interesting, since they illustrate the scalability issues better. The statistics

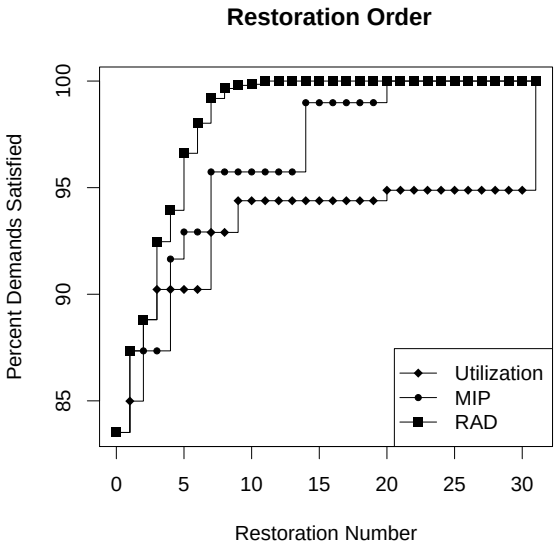

Figure 4.3: Restoration Plans on Benchmark 20.

indicate that the RAD algorithm consistently outperforms the MIP approach, improving the solution quality from 2.716 to 2.988 on average. The MIP approach also has severe difficulties on the larger instances. The detailed data reveals that the RAD algorithm usually matches the optimal solutions and, in cases where optimal solutions are not obtained, it often improves on the (suboptimal) MIP solution. The results also show that the RAD algorithm is significantly faster than the MIP approach.

The MRSP stage significantly reduces the damage set $|D|$ (close to a factor of 2). The results with the MRSP (the last three columns of Tables 4.1 and 4.2) indicate the MRSP brings significant improvements to the MIP model. The average quality on the smaller benchmarks is improved from 2.716 to 2.981 and seven more benchmarks become feasible. The runtime benefits to the MIP model are also significant, as 10 more instances can be proven optimal and the proof runtimes are reduced by a factor of 10. The quality improvements of the MRSP to the RAD algorithm are negligible, except for the largest benchmarks. For the largest benchmarks, the MRSP increases solution quality from 2.23 to 2.389. The MRSP also produces runtime benefits: the RAD algorithm terminates early on 29 benchmarks and the average early completion time is reduced by a factor of five, to less than five minutes.

The decoupling of the ROP problem into the MRSP+ROP problems may remove the optimal ROP solution, as Benchmark 5 indicates. However such effects become insignificant as the damage size grows and the challenge of finding a high-quality ROP solution increases. The decoupling is thus valuable in terms of both solution quality and run times.

	ROP			MRSP+ROP		
BM	$\|D\|$	MIP	RAD	$\|R\|$	MIP	RAD
1	10	1.42*	1.42	3	1.42*	1.42
2	12	4.594*	4.594	6	4.594*	4.594
3	14	2.277*	2.277	7	2.277*	2.277
4	14	2.967*	2.967	6	2.967*	2.967
5	14	2.928*	2.928	6	2.852*	2.852
6	14	2.856*	2.856	6	2.856*	2.856
7	15	2.056*	2.056	8	2.039*	2.039
8	17	1.536*	1.536	12	1.536*	1.536
9	17	1.567	1.57	11	1.57*	1.57
10	19	–	1.739	9	–	1.739
11	20	–	1.058	13	–	1.058
12	23	1.818	1.882	10	1.849*	1.849
13	23	5.191	6.724	6	6.721*	6.721
14	24	7.392	7.392	4	7.392*	7.392
15	26	–	1.345	13	1.333*	1.333
16	26	1.66	1.956	10	1.953*	1.953
17	27	–	2.507	13	–	2.486
18	29	1.355	2.342	12	2.379*	2.379
19	31	–	1.595	18	1.595	1.595
20	31	2.155	3.52	8	3.585*	3.585
21	33	–	2.333	17	1.547*	2.136
22	36	1.678	1.796	13	1.711	1.799
23	43	–	2.283	24	–	2.356
24	46	–	2.49	28	2.163	2.558
25	49	–	2.029	22	2.076*	2.076
26	56	–	2.596	28	2.495	2.593
27	57	–	2.622	29	–	2.49
28	61	–	2.38	25	1.788	2.45
29	73	–	2.628	40	–	2.57
30	79	–	2.234	39	–	3.189
31	92	–	1.761	55	–	2.015
32	92	–	1.968	52	–	2.255
33	120	–	1.536	73	–	1.722
MIP-μ		2.716	2.988		2.981	2.987
MSRP MIP-μ			2.721		2.639	2.719
RAD-μ			2.513			2.558
Large RAD-μ			2.23			2.389

Table 4.1: Multiplicative Effects of ROP Algorithms.

4.8 Discussion

This chapter considered the restoration of multiple interdependent infrastructures after man-made or natural disasters. It presented a scalable approach to the last-mile restoration of the joint electrical power and gas infrastructures, which features complex cyclic interdependencies. The underlying algorithms build on the generic three-stage decomposition (Figure 1.2) that decouples the restoration ordering and the routing aspects. At the technical level, the key contributions of this chapter

	ROP			MRSP+ROP		
BM	$\|D\|$	MIP	RAD	$\|R\|$	MIP	RAD
1	10	178.8	147.6	3	5.192	26.47
2	12	503.5	538.2	6	46.11	454.6
3	14	214.7	160.1	7	33.65	59.94
4	14	153.8	134.1	6	13.32	38.17
5	14	402.2	127.5	6	41.43	45.71
6	14	570.7	468.4	6	37.03	233.3
7	15	1101	850.4	8	135.7	243.5
8	17	496.9	250.4	12	129	142.9
9	17	3614	1224	11	502.1	428.3
10	19	–	880.7	9	–	264.9
11	20	–	885.9	13	–	577.6
12	23	3619	1142	10	408.5	354.1
13	23	3619	1276	6	22.54	141.5
14	24	3615	435.3	4	7.029	57.8
15	26	–	864	13	763.5	389
16	26	3618	740.2	10	180.9	469.9
17	27	–	2007	13	–	529.1
18	29	3620	1739	12	1102	506.7
19	31	–	1782	18	3615	713.7
20	31	3626	1047	8	52.22	84.65
21	33	–	1249	17	207.6	466.1
22	36	3632	901.1	13	3615	201
23	43	–	3646	24	–	2481
24	46	–	3708	28	3618	1983
25	49	–	3446	22	1425	484.4
26	56	–	3693	28	3618	1874
27	57	–	3647	29	–	1784
28	61	–	2611	25	3620	517.9
29	73	–	3691	40	–	3685
30	79	–	3722	39	–	2965
31	92	–	3730	55	–	3682
32	92	–	3679	52	–	3679
33	120	–	3758	73	–	3709
Proved		8			18	
ROP Proof Time μ		452.7			55.18	
Early Finish			24			29
Finish Time μ			1038			256.3

Table 4.2: ROP Algorithm Runtimes (Seconds).

are mixed-integer programming models for finding a minimal restoration set, restoration ordering, and a randomized adaptive decomposition scheme that obtains high-quality solutions within the required time limits. The approach is validated on a large selection of benchmarks based on the U.S. infrastructures and state-of-the-art weather and fragility simulation tools. The results show significant improvements over current field practices.

Chapter 5

Accuracy of the Linearized DC Power Model

Restoring a power system after a significant disruption (e.g., a cascading blackout or a seasonal hurricane) is an important task with consequences for both human and economic welfare. Power system components must be repaired and then re-energized without causing additional network instability. The restoration effort should be prioritized to minimize the size of the blackout by jointly optimizing repairs and power restoration. Chapters 3 and 4 resolved these challenges by using a sequence of optimization problems based on the linearized DC model.

Power restoration problems are daunting for a variety of reasons. First, since no *typical* operating base point is known, solving the resulting AC power flow problems is often challenging [80]. Second, good restoration plans jointly optimize the routing of repair crews and the scheduling of component energizing. The resulting optimization is a mixed integer nonlinear programming problem that is extremely challenging computationally. As a consequence, the power restoration algorithms of Chapter 3 and 4 use the linearized DC model in several steps, which makes it possible to model the problem in terms of mixed integer programs instead of mixed integer nonlinear programs.

The linearized DC model has been adopted as a general-purpose tool in a variety of power system optimizations in recent years (e.g., [80, 91]). However, its accuracy has been the topic of much discussion: most papers (e.g., [80, 87]) take an optimistic outlook, while others (e.g., [94, 26]) are more pessimistic. This issue is of particular interest for power restoration that involves human and economic welfare. It is critical that the linearized DC solution be a reasonable approximation of a high-quality AC power flow solution to avoid causing additional network instability.

This chapter studies the adequacy of using the linearized DC model for power restoration. It shows that, for power restoration, the linearized DC model may underestimate line loading significantly and produce solutions that are not feasible in an AC solver. Moreover, the experimental results suggest that large line phase angles are a good indicator of inaccurate active and apparent

power estimations. An Angle-Constrained DC Power Flow (ACDCPF) model that enforces constraints on the line phase angles and can shed load and generation across the network is proposed as a solution. The practicality of the approach is demonstrated on more than 11,000 damage contingencies in the IEEE30 network and validated on real-world power restoration problems arising in disaster management. The results show that the ACDCPF model produces solutions that are highly correlated with the AC power flow model and that these improvements in accuracy come at a reasonably small cost in load shedding. In particular, in the restoration context, the ACDCPF model is shown to be much more reliable and to produce significant reductions in the size of the blackouts compared to the linear DC model.

The rest of the chapter is organized as follows, Section 5.1 gives a brief review on power system modeling. Section 5.2 motivates the chapter through a short review of the restoration order problem (see Section 3.7 for a detailed discussion) and Section 5.3 investigates the accuracy of the linearized DC model under multiple line outages. Section 5.4 discusses the phase angle constraints and their impact. Section 5.5 presents our ACDCPF model and its empirical evaluation. Section 5.6 discusses several subtle details of the study and the core implications of the results.

5.1 Power System Modeling — A Brief Review

Nomenclature Before discussing the details of the power flow equations, some basic notation and terms need to be introduced. A power network $\mathcal{PN} = \langle N, L \rangle$ is characterized by a collection of buses N and lines L. In an AC power system each bus has a voltage $\widetilde{V}$ and power $\widetilde{S}$. Both values are complex numbers, as indicated by the tilde. AC voltages are most often presented in a polar form $\widetilde{V} = |\widetilde{V}|\angle\theta^\circ$, where $|\widetilde{V}|$ is the voltage magnitude and θ° the phase angle. AC power is most often presented in a rectangular form $\widetilde{S} = p + iq$, where p is called the active power and q the reactive power. The power magnitude $|\widetilde{S}|$, called apparent power, is also of interest. Each line in the power system connects two buses n, m and is characterized by several properties. Impedance $\widetilde{Z}(n,m) = r(n,m) + ix(n,m)$ captures the line's complex resistance and is composed of a real term r called resistance and an imaginary term x called reactance. Admittance, the inverse of impedance, $\widetilde{Z}^{-1}(n,m) = \widetilde{Y}(n,m) = g(n,m) + ib(n,m)$, is of particular interest and is composed of a real term g called conductance and an imaginary term b called susceptance. A line may also have a capacity value $c(n,m)$ that is a limit on the flow of apparent power on the line. The complex power and voltage of a line have similar names to buses but two indexes are used (i.e. $\widetilde{S}_{nm}, \widetilde{V}_{nm}, p_{nm}$). Last, the line admittances are often represented in a matrix $\widetilde{Y}_{bus}$, called an *admittance matrix* (which is similar to Laplacian Kirchhoff matrices). Due to the incorporation of transformers and admittance to ground values, the admittance matrix values may vary slightly from the line admittances. The notations $\widetilde{Y}_{bus}(n,m), g^y(n,m)$, and $b^y(n,m)$ are used to refer to the $\widetilde{Y}_{bus}$ admittance, conductance, and susceptance respectively.

AC Power Flow The ground truth in this work is the single-phase AC power model, which is widely accepted as a high-quality approximation of the steady-state behavior of real-world power flows. The single-phase AC power flow model uses Kirchhoff's current law and Ohm's law in the complex plane to define the active and reactive power injected at each power bus n, i.e.,

$$p_n = \sum_{m \in N} |\widetilde{V}_n||\widetilde{V}_m|(g^y(n,m)\cos(\theta_n^\circ - \theta_m^\circ) + b^y(n,m)\sin(\theta_n^\circ - \theta_m^\circ))$$
$$q_n = \sum_{m \in N} |\widetilde{V}_n||\widetilde{V}_m|(g^y(n,m)\sin(\theta_n^\circ - \theta_m^\circ) - b^y(n,m)\cos(\theta_n^\circ - \theta_m^\circ)).$$

These AC power flow equations can be solved by iterative solution techniques such as the Newton-Raphson method [9, 36, 50]. Convergence of these methods is not guaranteed and, when the system is heavily loaded, the solution space is riddled with infeasible low-voltage solutions that are useless in practice [95]. In fact, finding a solution to the AC power flow when a base-point solution is unavailable is often"maddeningly difficult" [80].

The Linearized DC Model The linearized DC model is derived from the AC power flow model through a series of approximations justified by operational considerations. In particular, it is assumed that (1) the susceptance is large relative to the impedance $|b(n,m)| \gg |g(n,m)|$; (2) the phase angle difference $\theta_n^\circ - \theta_m^\circ$ is small enough to ensure $\sin(\theta_n^\circ - \theta_m^\circ) \approx \theta_n^\circ - \theta_m^\circ$; and (3) the voltage magnitudes $|\widetilde{V}|$ are close to 1.0 and do not vary significantly. Under these assumptions, the AC power flow equations reduce to

$$p_n = \sum_{m \in N}^{n \neq m} b^y(n,m)(\theta_n^\circ - \theta_m^\circ). \tag{5.1}$$

From a computational standpoint, the linearized DC model is much more appealing than the AC model: It forms a system of linear equations that admit reliable algorithms. These linear equations can also be embedded into optimization frameworks for decision support in power systems [104, 80, 91, 44, 19, 31]. However, it is important to verify whether the assumptions of the linearized DC model hold for each application domain.

Implementation Choices The linearized DC model is so pervasive that authors often forget to mention important implementation details. Indeed, reference [94] recently demonstrated that small changes in the model formulation may have a significant impact on its accuracy. Moreover, there are conflicting suggestions about how the $\widetilde{Y}_{bus}$ matrix is derived (e.g., using $1/x$ or $-\Im(\widetilde{Z}^{-1})$) [68, 107, 82, 48]). Our goal is to make the AC and DC power models as similar as possible and our implementation reflects this choice. In particular, we use the same susceptance value $b^y(n,m)$ in the AC and DC models and adopt the $\widetilde{Y}_{bus}$ calculation described and implemented in MATPOWER [109].

By necessity, the AC solvers use a slack bus to ensure the flow balance on the network when the total power consumption is not known a priori (due to line losses, for instance). As a consequence,

the various DC models considered here also use a slack bus so that the AC and DC models can be accurately compared. It should be emphasized that the ACDCPF model proposed here does not need a slack bus: the only reason for the slack bus in the ACDCPF model is to allow meaningful comparisons between the DC and AC models. This issue is discussed at greater length in Section 5.6.

5.2 Power Restoration and Linearized DC Models

This study is motivated by investigating the joint repair and restoration of the transmission network after significant damages from a natural disaster. The goal is to schedule the repairs and to re-energize the electrical components in order to minimize the size of the blackout. This joint repair/restoration problem is extremely challenging computationally and is solved through a sequence of optimization models [104]. Several of the models in the sequence use a linearized DC model and it is important to investigate whether this is adequate in the presence of significant disruptions of the transmission network.

For the purpose of this study, it is sufficient to consider only one of the optimization problems proposed in [104]: the Restoration Order Problem (ROP). Conceptually, the ROP can be formulated as follows: A collection of D power system components has been disrupted and must be re-energized one at a time. The goal is to find a restoration order $d_1, d_2, d_3, \ldots$ for all components $d_i \in D$ in order to maximize the served load over time. In [104], the ROP is modeled as a generalization of optimal transmission switching (OTS) (e.g., [44, 56]). More precisely, the ROP is built from a collection of OTS models, one for each restoration step, that are connected together to optimize the restoration order globally.

Ideally, the ROP problem should be solved using an AC power flow model. However, simply finding an AC power flow solution for such disruptions can be quite challenging, since no base-point solution is available. Furthermore, ROP and OTS problems require discrete decision variables to indicate which components are energized, producing highly complex mixed integer nonlinear programming models that are beyond the capabilities of existing optimization technology. As a result, OTSs and ROPs are modeled in terms of linearized DC models. Note that there is a significant difference between the OTS and the ROP models. In OTS models, lines are switched from a current working base-point, while the ROP has no base-point because the disruption is arbitrary and extensive. As a result, it is not clear how accurate the resulting DC model is and whether it can be used in practice for restoration problems. This chapter addresses both questions.

To compare the DC and AC models for ROPs, we ignore the optimization process and focus on the solution returned, i.e., a restoration ordering. The quality of an ordering can be evaluated by a series of power flow calculations. Each step implements a change in the network topology since an additional component comes online and the power flow calculation gives the increase in served loads (or, equivalently, the reduction in the blackout). Therefore, to understand the accuracy of the linearized DC model in the restoration context, it is sufficient to study the accuracy of the linearized

Model 20 Linearized DC Power Flow (LDC).

Inputs:

$\mathcal{PN} = \langle N, L \rangle$ – the power network

b^y – susceptance from a $\widetilde{Y}_{bus}$ matrix

s – slack bus index

Variables:

$\theta_i^\circ \in (-\Re, \Re)$ – phase angle on bus i

Subject to:

$$\theta_s^\circ = 0 \tag{M20.1}$$

$$p_n = \sum_{m \in N}^{n \neq m} b^y(n,m)(\theta_n^\circ - \theta_m^\circ) \quad \forall n \in N \;\; n \neq s \tag{M20.2}$$

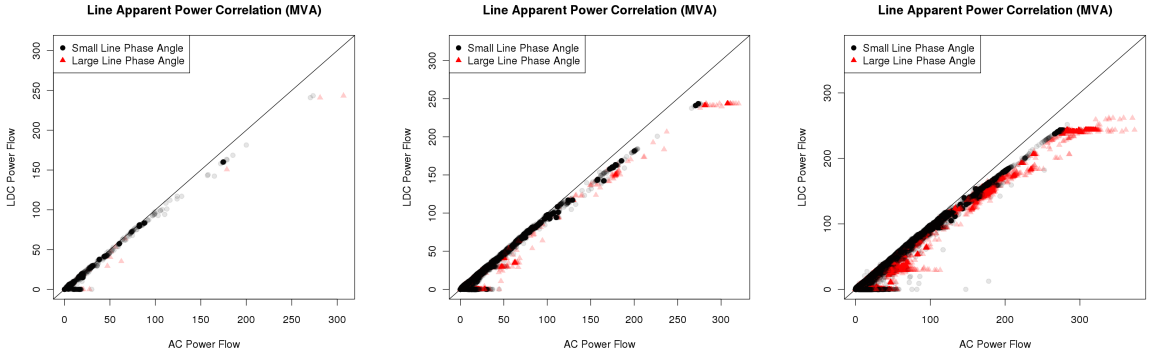

Figure 5.1: Accuracy of Apparent Power in N-1(left), N-2 (center), N-3 (right) Contingencies Using the Linearized DC Model.

DC model in isolation when it is subject to significant topological changes. Model 20 presents the linearized DC model (LDC) implementing our modeling assumptions. The model takes as inputs a power network $\mathcal{PN}$, susceptance values b^y, and the index s of the slack bus. The goal is to find the phase angles of all buses. Constraint M20.1 fixes the phase angle of the slack bus at 0 and constraint M20.2 implements the power flow model as defined in Equation 5.1. It is important to note that the power balance constraint is not posted for the slack bus. This allows the slack bus to pick up any unserved loads and balance the power in the network, as in a traditional AC power flow model.

5.3 DC Power Flow with Multiple Line Outages

To understand the accuracy of the linearized DC model under significant disruptions, we begin with a comprehensive empirical study of the IEEE30 system. We consider 11,521 damage contingencies, some with as many as three line outages (about 7% of the total network). Despite its ubiquity for optimization with N-1 contingency constraints, the accuracy of the linearized DC model has been evaluated only for specific application domains (e.g., [80, 26]) and under normal operating conditions

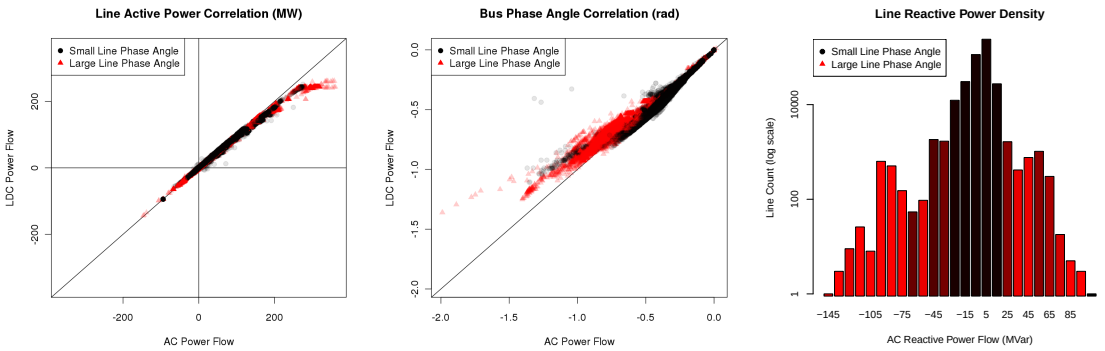

Figure 5.2: Accuracy Details of the N-3 Damage Contingencies Using the Linearized DC Model.

(e.g., [87, 94, 33]).[1] To our knowledge, this is the first direct study of DC model accuracy for N-1, N-2, and N-3 contingencies.

To compare AC and DC models, we measure the same information (e.g., active power, phase angles, ...) in both models and plot their correlation. Specifically, for some measurement data (e.g., the active power of a line), the x-axis gives the data value in the AC model and the y-axis gives the data value in the DC model. As a result, the closer the plot is to the line $x = y$, the better the AC and DC models agree. We focus primarily on apparent power since it is of particular interest to applications with line capacities. Obviously, in the DC model, apparent power is approximated by active power. The AC model is initialized with the voltages set to 1.0 and the phase angles to zero. For N-3 contingencies, it fails to converge in about 1% of the cases, as described below.

Figure 5.1 presents the correlation of apparent power for all N-1, N-2, and N-3 outages on the IEEE30 benchmark, giving us a total of 11,521 damaged networks. Each data point in the plots represents the apparent power of a line and, for brevity, the results are grouped by the number of outages and superimposed onto the same correlation graph. The plots also use red triangles for data points obtained from networks that feature *large* line phase angles (i.e., $|\theta_n^\circ - \theta_m^\circ| > \pi/12$). This makes it possible to understand the link between large line phase angles and discrepancies in apparent power.

Figure 5.1 highlights a number of interesting phenomena. First, observe that the overall accuracy of the model degrades significantly as the number of damaged components increases; this is of concern in power restoration applications. Second, the linearized DC model underestimates apparent power systematically and the more significant errors are almost always associated with large line phase angles. Finally, the plots indicate a general trend for the apparent power to lean to the right for large line loads. This is due to line losses that are not captured in the DC model. This limitation can be addressed in the linearized DC model as discussed in [80, 94, 21, 33], solution techniques that are completely orthogonal to the proposals in this chapter.

Figure 5.2 looks more deeply at these results and investigates the worst-case damage scenarios (i.e., N-3 contingencies) in more detail by presenting results for active power (left), bus phase angles

[1] One contingency case, where all lines loaded above 70% are removed, was considered in [94], but whether that case captures the general behavior of the linearized DC model under contingencies is unclear.

Line	AC Model $\theta_n^\circ - \theta_m^\circ$	AC Model p_{nm}	AC Model q_{nm}	LDC Model $\theta_n^\circ - \theta_m^\circ$	LDC Model p_{nm}
Normal Operation					
$1-2$	0.09338	173.2	-21.09	0.1023	160.1
$1-3$	0.1315	87.74	4.566	0.1479	83.28
Line 1-3 Damaged					
$1-2$	0.1478	270.4	-40.96	0.1556	243.4
$1-3$	–	–	–	–	–
Line 1-2 Damaged					
$1-2$	–	–	–	–	–
$1-3$	0.4862	304.0	43.00	0.4322	243.4

Table 5.1: Damage to Lines Connecting Bus 1 in the IEEE30 System.

(center), and reactive power (right). Once again, the color red represents large line phase angles. The left plot, depicting the correlation of active power, indicates that the linearized DC model underestimates active power on large line loads, which are also characterized by large line angles. The center plot depicts the correlation of the bus phase angles. It shows that the linearized DC model systematically underestimates the bus phase angles and the errors increase with large line phase angles. The right histogram depicts the number of lines whose reactive power fall within a certain range in the AC power flow. The color of each bar reflects the percentage of data points marked red in the other plots. The histogram reveals that N-3 contingencies produce a significant amount of reactive power on many lines, almost all of which exhibit large line phase angles. These results thus indicate that large line angles are correlated with under-approximations both of active power and reactive power, and hence produce significant errors in estimating apparent power.

To increase our intuition, it is also worthwhile to consider one particular bus from the IEEE30 system. Bus 1 in the IEEE30 is connected to buses 2 and 3 with impedances of $0.0192 + i0.0575$ and $0.0452 + i0.1652$ respectively (and thus susceptance values of -15.65 and -5.632 respectively). We investigate the N-1 contingencies around this bus to show how large angle differences and reactive flows are connected. Table 5.1 presents the results for three scenarios: normal operations, line 1-3 is damaged, and line 1-2 is damaged. Note that, in normal operations, two-thirds of the active power is flowing on line 1-2, so the contingency on that line is likely to be more interesting. The results indicate that, under normal operating conditions and when line 1-3 is damaged, the active power flows are very similar in both models and the phase angles are small. However, when line 1-2 is damaged, the phase angle approaches 0.5 radians coinciding with a large discrepancy in apparent power between the two models: the active flow in DC power model is 20% lower than the AC value and a large reactive flow exists.

In summary, the results show that the linearized DC model becomes increasingly less accurate under significant network disruptions. Many of these disruptions create large line phase angles, which lead to underestimations of active power and significant reactive power.

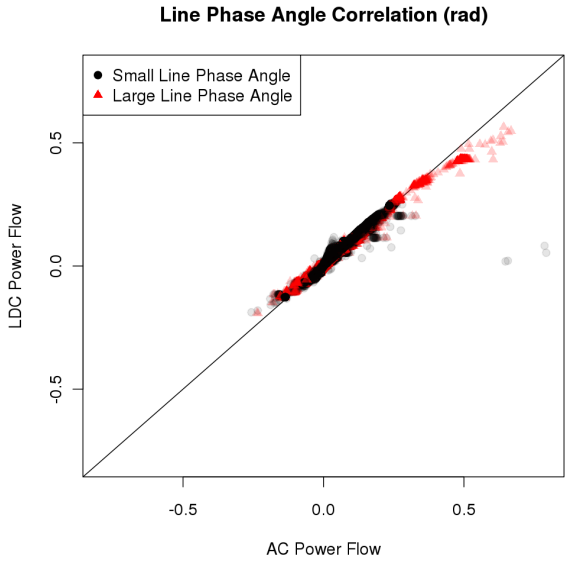

Figure 5.3: Accuracy of Line Phase Angles in N-3 Damage Contingencies Using the Linearized DC Model.

5.4 Constraints on Phase Angle Differences

The linearized DC model (Model 20) is a system of linear equations that can be solved very efficiently, particularly for sparse matrices, as is the case for power systems. However, since the phase angles are not restricted in the model, it potentially violates a fundamental assumption of the derivation, that $\sin(\theta_n^\circ - \theta_m^\circ) \approx \theta_n^\circ - \theta_m^\circ$. Moreover, the AC model guarantees that $-1 \leq \sin(\theta_n^\circ - \theta_m^\circ) \leq 1$ while the approximation of the sine term in the linearized DC model is unconstrained. This is problematic because the linearized DC model can produce solutions that are infeasible for the AC power model. Under normal operating conditions, this is not likely, but it is certainly possible when large disruptions occur. Obviously, it is possible to state the constraints $-1 \leq \theta_n^\circ - \theta_m^\circ \leq 1$ and use linear programming to obtain a feasible solution, but this does not guarantee an accurate approximation (the error is about 20% at the extremes). The linearized DC model will underestimate even reasonable phase angles, as shown in Figure 5.3. High model accuracy requires much stronger constraints.

5.5 Angle-Constrained DC Power Flow

This section proposes an Angle-Constrained DC Power Flow (ACDCPF) model that addresses the limitations discussed in the previous section and is particularly appropriate for power restoration. It is based on three key ideas:

1. Impose constraints on the line phase angles to avoid the power underestimations of the linearized DC model;

2. Use load and generation shedding to ensure accuracy of the model;

3. Use an objective function to maximize the served load.

The ACDCPF model is the linear program depicted in Model 21. The model receives as inputs the power network $\mathcal{PN}$, the susceptance values b^y, the slack bus index s, and the maximum generation G_i and a desired load L_i for each bus i. These last inputs are implicit in the linearized DC model since p_i is always equal to $G_i - L_i$. Its decision variables are the traditional bus phase angles θ_i°, as well as new decision variables g_i and l_i that represent the amount of generation and load at each bus i. The objective function (M21.1) maximizes the served load and hence the ACDCPF model sheds load only to ensure feasibility. Constraint (M21.2) models Kirchhoff's Current Law and ensures flow conservation for every bus. Constraint (M21.3) enforces the phase angle constraints to remedy the accuracy issues of the linearized DC model. Constraint (M21.4), which fixes the angle of the slack bus, is not necessary for the ACDCPF model in practice and is introduced here only so as to make meaningful comparisons between the ACDCPF and AC models. Note also that constraint (M21.3) can be posted for every bus, because generator dispatching and load shedding are used to balance load and generation. The ACDCPF model is close to Optimal Power Flow (OPF) models that support flexible generation but typically not load shedding [67, 6, 39]. Our experimental results indicate that the difference in phase angles should be no more than $\pm\pi/12$ (15 degrees) to ensure high accuracy. This tight constraint introduces no more than 1.1% error in active flow on each line due to the sine approximation. The exact value for this constraint is context-dependent. Observe that line phase angle constraints are very different from the bus phase angle constraints (e.g. $-\pi/6 \leq \theta^\circ \leq \pi/6$) employed in [104, 31, 44, 56]. Similar *branch angle constraints* are supported by MATPOWER [109] and appear in security-constrained applications [66]. However, in these contexts the phase angle constraints implement operational side-constraints and are not used to ensure model accuracy or shed loads. Adopting the ACDCPF model is a significant departure from previous work on the accuracy of the linearized DC models (e.g. [87, 94, 33]), which do not consider re-dispatching generation and shedding loads in the interest of model accuracy.

It is important to emphasize that, in power restoration, the ACDCPF model is not actually performing load shedding: Rather, it decides how much load can be served after a component has been repaired without exacerbating the instability of the network.[2]

5.5.1 Case Study: The IEEE30 Network

Section 5.3 showed that the accuracy of the linearized DC model may degrade with significant line damages and that large line phase angles are indicative of such degradation. This section repeats the experiments with the ACDCPF model. Since the ACDCPF model may shed load and generation, it is important to formulate the AC solver appropriately for comparison purposes. In particular, we use the active power values obtained from the ACFCPF model and we obtain the reactive power

[2]Load shedding may not be acceptable in settings where the load must be fully served. However, the ACDCPF may see other interesting uses with the advent of demand response.

Model 21 Angle-Constrained DC Power Flow (ACDCPF).

Inputs:

$\mathcal{PN} = \langle N, L \rangle$	– the power network
b^y	– susceptance from a $\widetilde{Y}_{bus}$ matrix
s	– slack bus index
G_i	– maximum generation at bus i
L_i	– desired load at bus i

Variables:

$\theta_i^\circ \in (-\Re, \Re)$	– phase angle on bus i
$g_i \in (0, G_i)$	– generation at bus i
$l_i \in (0, L_i)$	– load at bus i

Maximize:

$$\sum_{n \in N} l_n \tag{M21.1}$$

Subject to:

$$g_n - l_n = \sum_{m \in N}^{n \neq m} b^y(n,m)(\theta_n^\circ - \theta_m^\circ) \quad \forall n \in N \tag{M21.2}$$

$$-\pi/12 \leq \theta_n^\circ - \theta_m^\circ \leq \pi/12 \quad \forall \langle n, m \rangle \in L \tag{M21.3}$$

$$\theta_s^\circ = 0 \tag{M21.4}$$

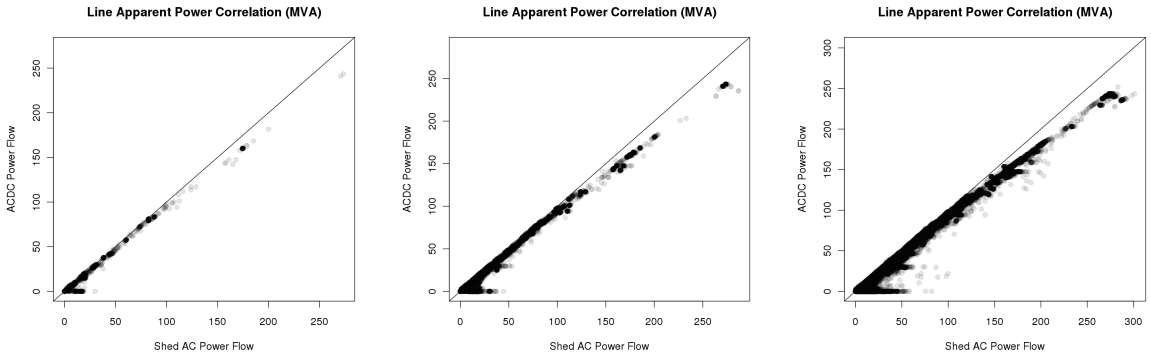

Figure 5.4: Accuracy of Apparent Power in N-1(left), N-2 (center), N-3 (right) Contingencies Using the ACDCPF Model.

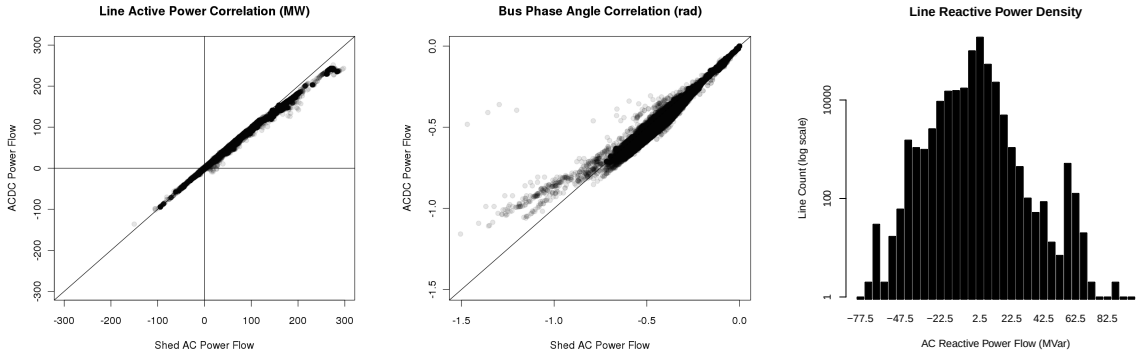

Figure 5.5: Accuracy Details of the N-3 Damage Contingencies Using the ACDCPF Model.

Damage	Contingencies	Linear. DC Solved	ACDCPF Model Solved	ACDCPF Model μ(Active Shed)
N-1	41	41	41	0.86%
N-2	820	819	820	2.10%
N-3	10,660	10,602	10,638	3.73%

Table 5.2: AC Solvability of IEEE30 Damage Contingencies.

values by scaling the active power using the power factor, thus shedding active and reactive power proportionally. This shedding approach provides reasonable reactive voltage support for a wide range of networks.

Figure 5.4 revisits the correlation of apparent power under different damage conditions. The figure depicts the N-1, N-2, and N-3 damage contingencies on the IEEE30 system and indicates that the ACDCPF model is performing very well. The errors are now mainly due to line losses on heavily loaded lines. As remarked earlier, corrections for line losses are well studied and can be integrated in the ACDCPF model (e.g., [80, 94, 33, 21]). There is only one outlier in 10,638 solved N-3 contingencies, which is rather promising. Figure 5.5 dives into the worst-case N-3 damage scenarios and presents results for active power (left), bus phase angles (center), and reactive power (right). The figure now shows a strong correlation for active power and a significant reduction in reactive power. The bus phase angles also show significantly better correlation.

These results indicate that line phase angle constraints largely remedy the inaccuracies of the linearized DC model and stabilize reactive power flows. However, this is at the cost of load shedding and it is important to quantify this loss. Table 5.5.1 summarizes the average load shedding for each contingency class. The results indicate that the loss in active power is reasonable: it is about 2% for N-2 contingencies and about 4% for N-3 contingencies on average. The table also reports how many times the ACDCPF solution can be transformed into a feasible AC solution and contrasts that result with the linearized DC model. The results show that the ACDFPF model leads to an AC solution 99.8% of the time and solves 30 more contingencies than the linearized DC model.

5.5.2 Case Study: Power Restoration

We now investigate the ACDCPF model for power restoration. In this context, the damage is so extensive that the full load cannot be served and load shedding always takes place. Figure 5.6 investigates a hurricane disaster scenario based on the United States power infrastructure. Additional scenarios are omitted for space considerations. In all graphs, two restoration plans are compared, an unconstrained (DCPF RP) and an angle-constrained plan (ACDCPF RP). The x-axis indicates how the power flow solution changes as new restoration actions are executed.

The DC restoration timeline (left) shows the restoration plans proposed by the DCPF and ACDCPF models. The ACDCPF RP restores about 10% less power because it sheds load to satisfy the angle constraint.

The AC restoration timeline (center) is much more illuminating. It depicts the restoration plans

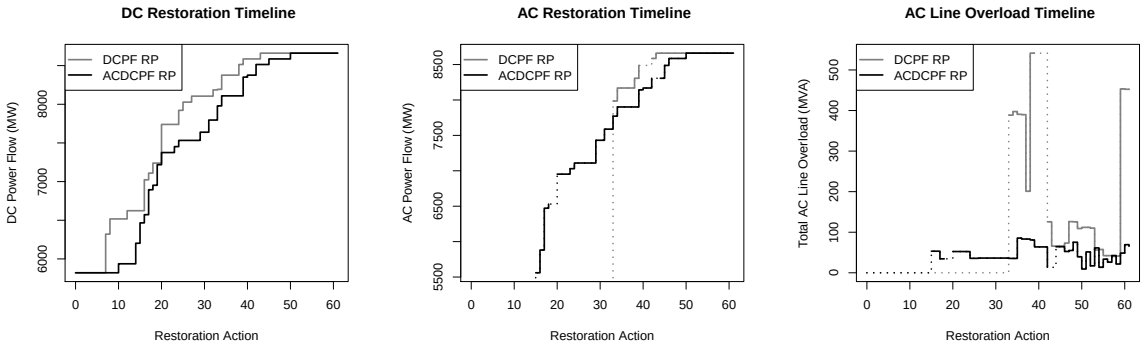

Figure 5.6: The Restoration Order Problem With and Without Angle Constraints and the Impacts on AC Power Flow.

obtained by the AC model when initialized with the DC solutions. First, observe that a number of data points are missing. This is caused by the inability of the AC solver to converge, which is not surprising given that there is no hand-tuning of the AC model and, as mentioned above, solution convergence under unfamiliar conditions is considered "maddeningly difficult" [80]. The dotted lines compensate for the missing data by adopting the previous power flow solution when available. The AC restoration timeline thus demonstrates that the DCPF model does not easily admit an AC solution under significant disruptions, while the ACDCPF model consistently produces solutions. In addition, and more importantly, the AC restoration plan for the ACDCPF model is very close to its DC counterpart, while the DCPF model produces a substantially worse AC restoration plan.

The right-hand graph in Figure 5.6 shows the total apparent power that exceeds all of the line capacities at various stages of restoration. Indeed, the IEEE30 case study indicates that large phase angles coincide with significant under-approximations of apparent power and that phase angle constraints mitigate these effects. The right-hand figure confirms these results for the ROP by investigating the amount of power that exceeds line capacities. Less than half of the data points for the DCPF model are available, but those data points can have staggering line overloads, reaching up to 500 MVA. On the contrary, the ACDCPF model significantly reduces line overloads and ensures they are consistently under 100 MVA.

Together, these results indicate that the ACDCPF model makes significant improvements in power restoration. Although its DC restoration plan restores 10% less power over time, its AC restoration plan restores significantly more power and keeps line overloads under control, which the DCPF model does not.

5.6 Discussion

5.6.1 Impact of the Slack Bus

The power system slack bus is a modeling artifact for solving the AC power flow equations and does not exist in power system operations [96]. In fact, the slack bus formulation is undesirable

as it accumulates line losses along the path to the slack bus, increases the load at the slack bus, and potentially causes larger phase angles and inaccuracies in apparent power [94]. It is important to emphasize that the ACDCPF does not require a slack bus: *The ACDCPF model is capable of shedding load and generation appropriately to balance the network without resorting to a slack bus or dedicated heuristics.* The slack bus was used here only to allow natural comparisons between the DC and AC models. Model inaccuracies may be overestimated by the slack-bus-based AC formulation because it is an approximation of real power system behavior. From an analytic standpoint, it would be interesting to compare the ACDCPF model with an AC solver using slack bus distribution. Such an AC solver models the real power systems more accurately and provides a better basis for comparison to the ACDCPF model since it can shed load and generation at all buses.

5.6.2 Line Capacities

It should be clear that line capacities can easily be added to the ACDCPF model, which is a linear program. In fact, the ACDCPF model for power restoration uses capacity constraints on the lines. Line capacities have an interesting connection to line phase angle constraints. In an AC model, a line capacity $c(n, m)$ imposes the constraint

$$\sqrt{p_{nm}^2 + q_{nm}^2} \leq c(n, m)$$

which, in linearized DC models, simplifies to

$$-c(n, m) \leq p_{nm} \leq c(n, m).$$

Expanding the definition of p_{nm} (Equation 5.1) and dividing by the susceptance $b(n, m)$ gives

$$-\frac{c(n, m)}{b(n, m)} \leq \theta_n^\circ - \theta_m^\circ \leq \frac{c(n, m)}{b(n, m)}.$$

Therefore, line capacity constraints in the ACDCPF model can be viewed as line phase angle constraints. In practice, line capacity constraints may be more or less restrictive than line phase angle constraints and there is no harm in posting both constraints in a linear program. Interestingly, for the standard IEEE benchmarks and those provided with MATPOWER, capacity constraints are significantly less constraining than a $\pm\pi/12$ phase angle constraint. The relationship between line angle constraints and line capacity constraints is fortunate because many existing power system optimization tools support line capacity constraints. A simple preprocessing step can transform line phase angle constraints into equivalent line capacity constraints to ensure model accuracy in existing power system optimization tools.

5.6.3 Integrating Line Losses

The results of the ACDCPF model from Section 5.5.1 show significant accuracy improvements. However, there is a tendency for the large power flows to be underestimated, as indicated by the data points leaning to the right. The source of this type of error was identified in [33], as the omission

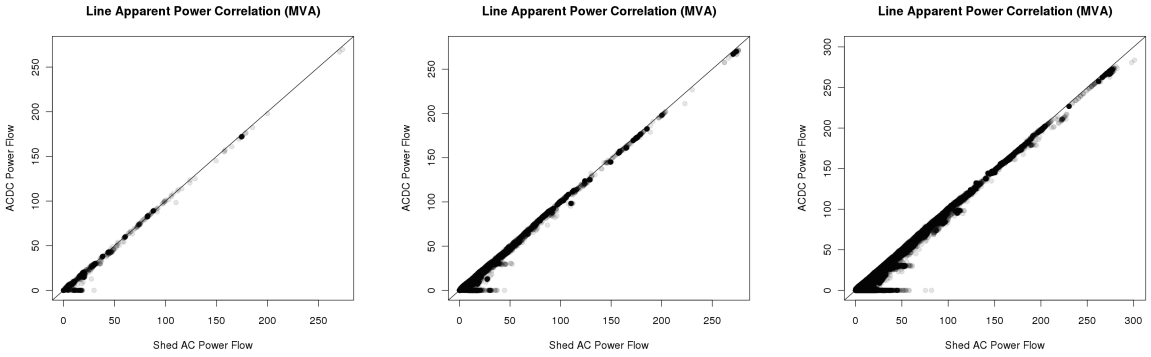

Figure 5.7: Accuracy of Apparent Power in N-1 (left), N-2 (center), N-3 (right) Contingencies Using the ACDCPF-LL Model.

Figure 5.8: Accuracy Details for the N-3 Damage Contingencies Using the ACDCPF-LL Model.

of line losses in the linearized DC model. Fortunately, [33] also proposed an extension to incorporate line losses in the linearized DC model. These extensions are easily integrated into the ACDCPF model to produce an ACDCPF-LL model. Figures 5.7 and 5.8 repeat the experiments of Section 5.5.1 with the ACDCPF-LL model. They confirm that line losses were the source of inaccuracy and illustrate that ACDCPF-LL is a robust and highly accurate linear model.

5.6.4 Applicability to Power Restoration

The experimental results indicate that the ACDCPF model remedies many of the limitations of the linearized DC model, provides highly accurate approximation of active power, and results in solutions with significantly less reactive power. Moreover, the load shedding performed by the ACDCPF model is small, about 4% on average for N-3 contingencies on the IEEE30 system and 10% for power restoration problems on the U.S. infrastructure for realistic hurricane scenarios. The experimental results show that the ACDCPF model is much more reliable and produces a significant reduction in the blackout area compared to the DCPF model. Furthermore, additional accuracy may be gained by integrating lines losses, as in the ACDCPF-LL model.

These results indicate that a traditional linearized DC model may be inappropriate for applications with significant topological changes, such as disaster management. However, the extensions of the ACDCPF model and the ACDCPF-LL model are resonable in such a context. It is important to note that all the models presented in Chapter 3 and 4 and may easily be updated to the ACDCPF-LL model for increased accuracy.

Chapter 6

Conclusions

Through three case studies — distribution of relief supplies, restoration of the power grid, and restoration of interdependent natural gas and power networks — this work has demonstrated that novel hybrid optimization techniques can improve the current best practices in disaster management within the aggressive runtime constraints. The key to the solutions' success is problem decomposition and utilizing the most advantageous optimization technology on each subproblem. The technical contributions of this work include: formalizing a number of complex disaster recovery problems; building models that effectively solve those problems; and developing a general framework (Figure 1.3) for solving a broad class of disaster preparation and recovery problems. The contributions touch on the fields of disaster management, humanitarian logistics, power systems engineering, stochastic optimization, and location routing.

The approaches were validated using real-world benchmarks provided by Los Alamos National Laboratory. These benchmarks were constructed using the U.S. infrastructure and state-of-the-art hazard and fragility simulation tools. Many components of this work are currently deployed at Los Alamos National Laboratory as part of the National Infrastructure Simulation and Analysis Center (NISAC) and are being used to aid federal organizations, such as the Department of Energy and the Department of Homeland Security, in preparing for and responding to disasters. The tools have been integrated into NISAC's *fast-response* center, which is activated any time a hurricane over category two is projected to land on U.S. soil.

Although this work significantly improves current field practices, computational models are by their nature only approximations of real-world systems. In the future, this work should be extended to incorporate more detailed models of reality. Some possible directions for improving the models are: increasing the number of interdependent infrastructures considered for restoration; broadening the study of different threat types to non-hurricane disaster scenarios; increasing the fidelity of the infrastructure models to include nonlinear or transience considerations; exploring the Pareto frontier of various restoration plans to understand the multi-objective tradeoffs; and enriching the stochastic model to allow probabilistic damage in the disaster scenarios. All of these are interesting and orthogonal directions for further investigation. The impact of these extensions remains unclear

and such tools should be compared to this work to understand the value of richer computational models.

Reflecting more broadly, every model in this work demonstrates that problem-driven decompositions and local search methods produce an excellent tradeoff between run time and solution quality. Decompositions and local search merely approximate a problem's optimal solution and applying both simultaneously, as this work does, could have compounding negative effects. Surprisingly, such effects are negligible and the decomposed models produce the best known solutions to many of the problems studied here. This result suggests that, as the scale or combinatorics of a problem become overwhelming, decomposing the problem and solving each piece independently with local search is the best solution approach; producing high quality solutions in a relatively short amount of time. This work demonstrates such a property in the domain of disaster management but I conjecture that it holds in other domains that combine several classic combinatorial optimization problems or have significant scalability requirements. The recent emergence and success of randomized adaptive decompositions [16, 17, 81] provides evidence of such a trend.

Appendix A

Publication Summary

Several components of this thesis first appeared in conference proceedings. The contributions of Chapter 2 first appeared in the Seventh [102] and Eighth [34] International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR). The contributions of Chapter 3 have appeared in the Second International Conference on Vulnerability and Risk Analysis and Management (ICVRAM) [30], the IEEE Power and Energy Society General Meeting (PES) [31], and the Seventeenth International Conference on Power Systems Computation (PSCC) [104]. The contributions of Chapter 4 appeared in the Twenty-Sixth Conference on Artificial Intelligence (AAAI) [35]. The contributions of Chapter 5 appeared in the IEEE Power and Energy Society General Meeting (PES) [32] and leverage contributions of a related work at the same venue [33] (a detailed discussion of [33] is outside the scope of this document).

Bibliography

[1] M. Adibi. *Power System Restoration (Methodologies and Implementation Strategies)*. Wiley-IEEE Press, 2000.

[2] M.M. Adibi and L.H. Fink. Power system restoration planning. *IEEE Transactions on Power Systems*, 9(1):22–28, 1994.

[3] M.M. Adibi, L.R.J. Kafka, and D.P. Milanicz. Expert system requirements for power system restoration. *IEEE Transactions on Power Systems*, 9(3):1592–1600, 1994.

[4] U.S. Federal Emergency Management Agency. HAZUS Overview. Available online at http://www.fema.gov/plan/prevent/hazus/, 2010.

[5] M. Albareda-Sambola, J. A. Diaz, and E. Fernandez. A compact model and tight bounds for a combined location-routing problem. *Computer & Operations Research*, 32:407–428, 2005.

[6] O. Alsac, J. Bright, M. Prais, and B. Stott. Further developments in LP-based optimal power flow. *IEEE Transactions on Power Systems*, 5(3):697–711, 1990.

[7] J.J. Ancona. A framework for power system restoration following a major power failure. *IEEE Transactions on Power Systems*, 10(3):1480–1485, 1995.

[8] B. Balcik, B. Beamon, and K. Smilowitz. Last-mile distribution in humanitarian relief. *Journal of Intelligent Transportation Systems*, 12(2):51–63, 2008.

[9] Ross Baldick. *Applied Optimization: Formulation and Algorithms for Engineering Systems*. Cambridge University Press, 2009.

[10] Gulay Barbarosoglu, Linet Ozdamar, and Ahmet Cevik. An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. *European Journal of Operational Research*, 140(1):118–133, 2002.

[11] John Battelle. *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture*. Portfolio, 2005.

[12] B. Beamon. Humanitarian relief chains: issues and challenges. *34th International Conference on Computers & Industrial Engineering*, pages 77–82, 2008.

[13] Keith Bell, Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. The role of AI planning as a decision support tool in power substation management. *AI Commun.*, 22(1):37–57, 2009.

[14] R. Bent and P. Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 8(4):515–530, 2004.

[15] R. Bent and P. Van Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers and Operations Research (Special Issue on Applications in Combinatorial Optimization)*, pages 875–893, 2006.

[16] R. Bent and P. Van Hentenryck. Randomized adaptive spatial decoupling for large-scale vehicle routing with time windows. In *Proceedings of the 22th National Conference on Artificial Intelligence (AAAI'07)*. AAAI Press, 2007.

[17] Russell Bent and Pascal Van Hentenryck. Spatial, temporal, and hybrid decompositions for large-scale vehicle routing with time windows. In David Cohen, editor, *CP*, volume 6308 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2010.

[18] Piergiorgio Bertoli, Alessandro Cimatti, John K. Slaney, and Sylvie Thiébaux. Solving power supply restoration problems with planning via symbolic model checking. In Frank van Harmelen, editor, *ECAI*, pages 576–580. IOS Press, 2002.

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

[20] Eric Bonabeau. Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 3):7280–7287, 2002.

[21] A. Borghetti, M. Paolone, and C. A. Nucci. A mixed integer linear programming approach to the optimal configuration of electrical distribution networks with embedded generators. *Proceedings of the 17th Power Systems Computation Conference (PSCC'11), Stockholm, Sweden*, 2011.

[22] Sergey V. Buldyrev, Roni Parshani, Gerald Paul, H. Eugene Stanley, and Shlomo Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, 2010.

[23] Ann Melissa Campbell, Dieter Vandenbussche, and William Hermann. Routing for relief efforts. *Transportation Science*, 42(2):127–145, 2008.

[24] Rui Carvalho, Lubos Buzna, Flavio Bono, Eugenio Gutiérrez, Wolfram Just, and David Arrowsmith. Robustness of trans-european gas networks. *Phys. Rev. E*, 80:016106, 2009.

[25] B. Cavdaroglu, E. Hammel, J.E. Mitchell, T.C. Sharkey, and W.A. Wallace. Integrating restoration and scheduling decisions for disrupted interdependent infrastructure systems. *Annals of Operations Research (to appear)*, 2012.

[26] O. Ceylan, A. Ozdemir, and H. Dag. Branch outage solution using particle swarm optimization. In *Power Engineering Conference, 2008. AUPEC '08*, pages 1–5, 2008.

[27] Der-San Chen, Robert G. Batson, and Yu Dang. *Applied Integer Programming: Modeling and Solution*. Wiley, 2010.

[28] N. Cho. Critical infrastructure rebuild prioritization using simulation optimization. Master's thesis, Air Force Institute of Technology, 2007.

[29] Elvin Coban and John N. Hooker. Single-facility scheduling over long time horizons by logic-based benders decomposition. In Lodi et al. [71], pages 87–91.

[30] C. Coffrin, P. Van Hentenryck, and R. Bent. Strategic planning for power system restoration. *Proceedings of the International Conference on Vulnerability and Risk Analysis and Management (ICVRAM 2011)*, 2011.

[31] C. Coffrin, P. Van Hentenryck, and R. Bent. Strategic stockpiling of power system supplies for disaster recovery. *Proceedings of the 2011 IEEE Power & Energy Society General Meetings (PES 2011)*, 2011.

[32] C. Coffrin, P. Van Hentenryck, and R. Bent. Accurate Load and Generation Scheduling for Linearized DC Models with Contingencies. *Proceedings of the 2012 IEEE Power & Energy Society General Meetings (PES)*, 2012.

[33] C. Coffrin, P. Van Hentenryck, and R. Bent. Approximating Line Losses and Apparent Power in AC Power Flow Linearizations. *Proceedings of the 2012 IEEE Power & Energy Society General Meetings (PES)*, 2012.

[34] Carleton Coffrin, Pascal Van Hentenryck, and Russell Bent. Spatial and objective decompositions for very large scaps. In Tobias Achterberg and J. Christopher Beck, editors, *CPAIOR*, volume 6697 of *Lecture Notes in Computer Science*, pages 59–75. Springer, 2011.

[35] Carleton Coffrin, Pascal Van Hentenryck, and Russell Bent. Last-mile restoration for multiple interdependent infrastructures. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012. (to appear).

[36] Mariesa Crow. *Computational Methods for Electric Power Systems, Second Edition (Electric Power Engineering Series)*. CRC Press, 2009.

[37] George Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1998.

[38] A. Delgadillo, J.M. Arroyo, and N. Alguacil. Analysis of electric grid interdiction with line switching. *IEEE Transactions on Power Systems*, 25(2):633–641, 2010.

[39] H.W. Dommel and W.F. Tinney. Optimal power flow solutions. *IEEE Transactions on Power Apparatus and Systems*, PAS-87(10):1866–1876, 1968.

[40] Leonardo Dueas-Osorio and Isaac Hernandez-Fajardo. Flow-based reliability assessment of infrastructure systems. In *14th World Conference on Earthquake Engineering (14WCEE)*, volume 31, pages 157–167, 2008.

[41] Leonardo Dueas-Osorio and Srivishnu Mohan Vemuru. Cascading failures in complex infrastructure systems. *Structural Safety*, 31:157–167, 2009.

[42] Serhan Duran, Marco Gutierrez, and Pinar Keskinocak. Pre-Positioning of emergency items worldwide for CARE International. *Interfaces*, 41(3):223–237, 2011.

[43] Dynamic Decision Technologies (Dynadec). Comet 2.1 user manual. http://dynadec.com, 2009.

[44] E.B. Fisher, R.P. O'Neill, and M.C. Ferris. Optimal transmission switching. *IEEE Transactions on Power Systems*, 23(3):1346–1355, 2008.

[45] Filippo Focacci, Franois Laburthe, and Andrea Lodi, editors. *Handbook of Metaheuristics (International Series in Operations Research and Management Science)*. Springer, 2003.

[46] Filippo Focacci, Franois Laburthe, and Andrea Lodi, editors. *Handbook of Metaheuristics (International Series in Operations Research and Management Science)*, chapter Chapter 13, Local Search and Constraint Programming. Springer, 2003.

[47] Fritz Institute. Fritz Institute Website. http://www.fritzinstitute.org, 2008.

[48] Antonio Gomez-Exposito, Antonio J. Conejo, and Claudio Canizares. *Electric Energy Systems: Analysis and Operation (Electric Power Engineering Series)*. CRC Press, 2008.

[49] Jing Gong, Earl E. Lee, John E. Mitchell, and William A. Wallace. Logic-based multiobjective optimization for restoration planning. In Wanpracha Chaovalitwongse, Kevin C. Furman, and Panos M. Pardalos, editors, *Optimization and Logistics Challenges in the Enterprise*, volume 30, pages 305–324. Springer US, 2009.

[50] John Grainger and William Stevenson Jr. *Power System Analysis*. McGraw-Hill Science/Engineering/Math, 1994.

[51] P. Griffin, C. Scherrer, and J. Swann. Optimization of community health center locations and service offerings with statistical need estimation. *IIE Transactions*, 2008.

[52] Canan Gunes, Willem Jan van Hoeve, and Sridhar Tayur. Vehicle routing for food rescue programs: a comparison of different approaches. In Lodi et al. [71], pages 176–180.

[53] D. Gunnec and F. Salman. A two-stage multi-criteria stochastic programming model for location of emergency response and distribution centers. In *International Network Optimization Conference*, 2007.

[54] Tarik Hadzic and Henrik Reif Andersen. Interactive reconfiguration in power supply restoration. In *11th International Conference on Principles and Practice of Constraint Programming CP05*. Springer-Verlag GmbH, 2005.

[55] P. Hansen and N. Mladenovic. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer Academic Publishers, Norwell, MA, 1998.

[56] K.W. Hedman, R.P. O'Neill, E.B. Fisher, and S.S. Oren. Optimal transmission switching with contingency analysis. *IEEE Transactions on Power Systems*, 24(3):1577–1586, 2009.

[57] Marilyn M. Helms, Lawrence P. Ettkin, and Sharon Chapman. Supply chain forecasting: collaborative forecasting supports supply chain management. *Business Process Management Journal*, 6(5):392–406, 2000.

[58] Pascal Van Hentenryck. *Constraint satisfaction in logic progamming.* Logic programming. MIT Press, 1989.

[59] P. Hines, E. Cotilla-Sanchez, and S. Blumsack. Do topological models provide good information about vulnerability in electric power networks? *ArXiv e-prints*, 2010.

[60] Benjamin F. Hobbs, Michael H. Rothkopf, Richard P. O'Neill, and Hung po Chao. *The Next Generation of Electric Power Unit Commitment Models (International Series in Operations Research and Management Science).* Springer, 2001.

[61] J.A. Huang, L. Audette, and S. Harrison. A systematic method for power system restoration planning. *IEEE Transactions on Power Systems*, 10(2):869–875, 1995.

[62] J.A. Huang, F.D. Galiana, and G.T. Vuong. Power system restoration incorporating interactive graphics and optimization. *Power Industry Computer Application Conference, 1991. Conference Proceedings*, pages 216–222, 1991.

[63] Michael Huang, Karen Smilowitz, and Burcu Balcik. Models for relief routing: Equity, efficiency and efficacy. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):2–18, 2012.

[64] James P. Ignizio. A review of goal programming: A tool for multiobjective analysis. *The Journal of the Operational Research Society*, 29(11):1109–1119, 1978.

[65] David Joslin and David P. Clements. Squeaky wheel optimization. *J. Artif. Intell. Res. (JAIR)*, 10:353–373, 1999.

[66] A. Khodaei and M. Shahidehpour. Transmission switching in security-constrained unit commitment. *IEEE Transactions on Power Systems*, 25(4):1937–1945, 2010.

[67] D.S. Kirschen and H.P. Van Meeteren. Mw/voltage control in a linear programming based optimal power flow. *IEEE Transactions on Power Systems*, 3(2):481–489, 1988.

[68] Upton George Knight. *Power systems engineering and mathematics*. Pergamon Press Oxford, New York, 1972.

[69] E.E. Lee, J.E. Mitchell, and W.A. Wallace. Assessing vulnerability of proposed designs for interdependent infrastructure systems. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004.

[70] 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*, 37(6):1303–1317, 2007.

[71] Andrea Lodi, Michela Milano, and Paolo Toth, editors. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010, Bologna, Italy, June 14-18, 2010. Proceedings*, volume 6140 of *Lecture Notes in Computer Science*. Springer, 2010.

[72] Staci Matlock. Nm gas co.: Does not anticipate additional disruptions to natural gas service. http://www.santafenewmexican.com/Local 20News/Gas-shortage-cuts-heat-for-25-000, Febuary 2011.

[73] L. Michel, A. See, and P. Van Hentenryck. Transparent parallelization of constraint programming. *INFORMS Journal on Computing*, 21(3):363–382, 2009.

[74] F. Monforti and A. Szikszai. A monte carlo approach for assessing the adequacy of the european gas transmission system under supply crisis conditions. *Energy Policy*, 38(5):2486–2498, 2010.

[75] A.L. Morelato and A.J. Monticelli. Heuristic search approach to distribution system restoration. *IEEE Transactions on Power Delivery*, 4(4):2235–2241, 1989.

[76] H. Mori and Y. Ogita. A parallel tabu search based approach to optimal network reconfigurations for service restoration in distribution systems. *Proceedings of the 2002 International Conference on Control Applications*, 2:814–819, 2002.

[77] T. Nagata, H. Sasaki, and R. Yokoyama. Power system restoration by joint usage of expert system and mathematical programming approach. *IEEE Transactions on Power Systems*, 10(3):1473–1479, 1995.

[78] G. Nagy and S. Salhi. Nested heuristic methods for the location-routing problem. *Journal of Operational Research Society*, 47:1166–1174, 1996.

[79] Min Ouyang and Leonardo Dueas-Osorio. An approach to design interface topologies across interdependent urban infrastructure systems. *Reliability Engineering and System Safety*, 96(11):1462–1473, 2011.

[80] T.J. Overbye, Xu Cheng, and Yan Sun. A comparison of the AC and DC power flow models for LMP calculations. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004.

[81] Dario Pacino and Pascal Van Hentenryck. Large neighborhood search and adaptive randomized decompositions for flexible jobshop scheduling. In Toby Walsh, editor, *IJCAI*, pages 1997–2002. IJCAI/AAAI, 2011.

[82] Lynn Powell. *Power System Load Flow Analysis (Professional Engineering)*. McGraw-Hill Professional, 2004.

[83] Christian Prins, Caroline Prodhon, Angel B. Ruiz, Patrick Soriano, and Roberto Wolfler Calvo. Solving the capacitated location-routing problem by a cooperative lagrangian relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483, 2007.

[84] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2):509–533, 2008.

[85] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3):371–386, 2010.

[86] J. Puchinger, G. Raidl, and U. Pferschy. The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, 22(2):250–265, 2010.

[87] K. Purchala, L. Meeus, D. Van Dommelen, and R. Belmans. Usefulness of dc power flow for active power flow analysis. In *Power Engineering Society General Meeting, 2005. IEEE*, volume 1, pages 454–459, 2005.

[88] Dorothy A. Reed. Electric utility distribution analysis for extreme winds. *Journal of Wind Engineering and Industrial Aerodynamics*, 96(1):123–140, 2008.

[89] T. Sakaguchi and K. Matsumoto. Development of a knowledge-based system for power system restoration. *IEEE Transactions on Power Apparatus and Systems*, PAS-102(2):320–329, 1983.

[90] J. Salmeron, K. Wood, and R. Baldick. Analysis of electric grid security under terrorist threat. *IEEE Transactions on Power Systems*, 19(2):905–912, 2004.

[91] J. Salmeron, K. Wood, and R. Baldick. Worst-case interdiction analysis of large-scale electric power grids. *IEEE Transactions on Power Systems*, 24(1):96–104, 2009.

[92] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of Fourth International Conference on the Principles and Practice of Constraint Programming (CP'98)*, pages 417–431, 1998.

[93] Ben Simon, Carleton Coffrin, and Pascal Van Hentenryck. Randomized adaptive vehicle decomposition for large-scale power restoration. In Nicolas Beldiceanu, Narendra Jussien, and Éric Pinson, editors, *9th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'12)*, volume 7298 of *Lectures Notes in Computer Science*, pages 379–394, Nantes, France, 2012. Springer Verlag.

[94] B. Stott, J. Jardim, and O. Alsac. DC power flow revisited. *IEEE Transactions on Power Systems*, 24(3):1290–1300, 2009.

[95] Y. Tamura, H. Mori, and S. Iwamoto. Relationship between voltage instability and multiple load flow solutions in electric power systems. *IEEE Transactions on Power Apparatus and Systems*, PAS-102(5):1115–1125, 1983.

[96] Shiqiong Tong and K.N. Miu. Participation factor studies for distributed slack bus models in three-phase distribution power flow analysis. In *Transmission and Distribution Conference and Exhibition, 2005/2006 IEEE PES*, pages 92–96, 2006.

[97] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA, 2001.

[98] D. Tuzun and L. I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116:87–99, 1999.

[99] United States Government. *The Federal Response to Hurricane Katrina: Lessons Learned*, 2006.

[100] P. Van Hentenryck. *Constraint-Based Local Search*. MIT Press, Cambridge, Mass., 2005.

[101] P. Van Hentenryck and R. Bent. *Online Stochastic Combinatorial Optimization*. MIT Press, Cambridge, Mass., 2006.

[102] P. Van Hentenryck, R. Bent, and C. Coffrin. Strategic planning for disaster recovery with stochastic last mile distribution. In Lodi et al. [71], pages 318–333.

[103] P. Van Hentenryck, R. Bent, and E. Upfal. Online stochastic optimization under time constraints. *Annals of Operations Research (Special Issue on Stochastic Programming)*, 177:151–183, 2009.

[104] P. Van Hentenryck, C. Coffrin, and R. Bent. Vehicle routing for the last mile of power system restoration. *Proceedings of the 17th Power Systems Computation Conference (PSCC'11)*, 2011.

[105] W. A. Wallace, D. Mendona, E. Lee, J. Mitchell, and J. Chow. Managing disruptions to critical interdependent infrastructures in the context of the 2001 world trade center attack. *Beyond September 11: An account of post-disaster research*, pages 165–198, 2003.

[106] L. Van Wassenhove. Humanitarian aid logistics: supply chain management in high gear. *Journal of the Operational Research Society*, 57(1):475–489, 2006.

[107] Allen J. Wood and Bruce F. Wollenberg. *Power Generation, Operation, and Control*. Wiley-Interscience, 1996.

[108] M.H. Yolcu, Z. Zabar, L. Birenbaum, and S.A. Granek. Adaptation of the simplex algorithm to modeling of cold load pickup of a large secondary network distribution system. *IEEE Transactions on Power Apparatus and Systems*, PAS-102(7):2064–2068, 1983.

[109] R.D. Zimmerman, C.E. Murillo-Sandnchez, and R.J. Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems*, 26(1):12–19, 2011.