Robust Optimization for Network-based Resource Allocation Problems under Uncertainty

by

Lavanya Marla

Submitted to the Department of Civil and Environmental Engineering
and Operations Research Center
in partial fulfillment of the requirements for the degrees of
Master of Science in Transportation
and
Master of Science in Operations Research
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2007

Author .....
Department of Civil and Environmental Engineering
and Operations Research Center
May 24, 2007

Certified by .....
Cynthia Barnhart
Professor of Civil and Environmental Engineering
Co-Director, Operations Research Center
Thesis Supervisor

Accepted by .....
Dimitris Bertsimas
Boeing Professor of Operations Research
Co-Director, Operations Research Center

Accepted by .....
Daniele Veneziano
Chairman, Department Committee for Graduate Students

Robust Optimization for Network-based Resource Allocation Problems under Uncertainty

by

Lavanya Marla

Submitted to the Department of Civil and Environmental Engineering
and Operations Research Center
on May 24, 2007, in partial fulfillment of the
requirements for the degrees of
Master of Science in Transportation
and
Master of Science in Operations Research

Abstract

We consider large-scale, network-based, resource allocation problems under uncertainty, with specific focus on the class of problems referred to as multi-commodity flow problems with time-windows. These problems are at the core of many network-based resource allocation problems. Inherent data uncertainty in the problem guarantees that deterministic optimal solutions are rarely, if ever, executed. Our work examines methods of proactive planning, that is, robust plan generation to protect against future uncertainty. By modeling uncertainties in data corresponding to service times, resource availability, supplies and demands, we can generate solutions that are more robust operationally, that is, more likely to be executed or easier to repair when disrupted. The challenges are the following: approaches to achieve robustness 1) can be extremely problem-specific and not general; 2) suffer from issues of tractability; or 3) have unrealistic data requirements.

We propose in this work a modeling and algorithmic framework that addresses the above challenges. Our modeling framework involves a decomposition scheme that separates problems involving multi-commodity flows with time-windows into routing (that is, a routing master problem) and scheduling modules (that is, a scheduling sub-problem), and uses an iterative scheme to provide feedback between the two modules, both of which are more tractable than the integrated model. The master problem has the structure of a multi-commodity flow problem and the sub-problem is a set of network flow problems. This decomposition allows us to capture uncertainty while maintaining tractability. Uncertainty is captured in part by the master problem and in part by the sub-problem. In addition to solving problems under uncertainty, our decomposition scheme can also be used to solve large-scale resource allocation problems without uncertainty. As proof-of-concept, we apply our approach to a vehicle routing and scheduling problem and compare its solutions to those of other robust optimization approaches. Finally, we propose a framework to extend our robust,

decomposition approach to the more complex problem of network design.

Thesis Supervisor: Cynthia Barnhart
Title: Professor of Civil and Environmental Engineering
Co-Director, Operations Research Center

Acknowledgments

Writing this thesis helped me learn a lot about myself. I am grateful to have been given this wonderful opportunity at MIT to meet so many wonderful people and learn from them.

I owe heartfelt thanks to my advisor, Cynthia Barnhart, whose encouragement helped me to gain a fresh perspective on research as well as life. She has shown great care and interest in my work, giving my interests the first priority. Her dedication to her students is remarkable. She always has time for students, no matter how much other stuff she has going on at the same time. Thank you, Cindy, for being a mentor, a friend and an advisor, all rolled into one.

I would like to thank colleagues in the Transportation Students Group and Operations Research Center for their friendship, and to Maria Marangiello and Ginny Siggia for their assistance.

Thanks to my friends and support group, who are simply priceless. Thanks to Mythili and Madhu for supporting me at crunch time, Naveen and Vikram for their 'offline' encouragement, and Padma and Pavithra for being such wonderful friends.

Thanks to Mom and Dad for always being there, and Ramya for her trademark brand of comedy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Resource allocation involves the distribution and utilization of available resources across the system. Because resource availability is usually scarce and expensive, it becomes important to find optimal or even 'good' solutions to such problems. Thus, resource allocation problems represent an important class of problems faced by mathematical programmers.

Conventionally, such resource allocation problems have been modeled and solved assuming input data to be known with certainty. Such models that consider the inputs to be invariant are called *nominal models* and their solutions are denoted *nominal solutions*. However, in practice, these assumptions are rarely, if ever, true, which raises questions regarding the validity and practicability of the solutions obtained under these assumptions. The presence of such uncertainty can disrupt operations and cause plans to be infeasible during implementation. In fact, Mulvey [25] and Ben-Tal and Nemirovski [6] show that such nominal solutions can become irrelevant in the presence of real-world uncertainty.

In this thesis, we study resource allocation problems that are network-based, involving the flow of resources over a typically, large-scale network in an optimal manner.

This sub-class of problems has some applications that are of special interest, including those that arise in the areas of transportation, logistics and communications. Specific examples include airline scheduling, vehicle routing, service network design, load distribution, production planning, computer scheduling, portfolio selection, and apportionment. These application domains have important economic value, and high importance is attached to achieving efficient solutions.

Network-based resource allocation problems have several characteristics that contribute to their complexity. Optimization must be performed over both space and time dimensions, that is, resources need to be distributed in space and over time. The networks considered are usually very large, and it is important to find solution methods that are scalable. Multiple, heterogenous resources need to be allocated. Costs and constraints are often non-linear, and need to be approximated well to facilitate solution tractability while maintaining model realism. Moreover, discreteness of resources requires that solutions to these problems be integer or binary, which further complicates solving the mathematical program.

In addition, uncertainty caused due to inherent system *stochasticity* plays an important role. In the real world, randomness and probability are always at work, and no scenario can be predicted exactly. Thus planners usually work with nominal '*best-guess*' or '*mean-value*' system parameters. When the realized parameters are different from those planned, the system no longer behaves in a near-optimal way. Because the 'optimized' system is likely to have little slack, infeasibilities are encountered, and sometimes frequently. Hence, solutions produced under these conditions are rarely (if ever) executed, and certainly, never truly optimal. In this thesis, our objective is to build 'robust' network resource allocation solutions that:

1. are less fragile to disruption,

2. easier to repair if needed, and

3. optimize the *realized*, not just planned, problem objective.

The potential impact of providing *robust*, efficient resource allocations over networks can be enormous. For example, plans that were disrupted during operations cost the airline industry 6 billion dollars during the year 2006 [1].

In what follows, we first provide some background related to resource allocation problems, define the problem we solve, and discuss some challenges in this research.

1.2 Background

Resource allocation problems have been extensively researched. In generalized optimization terminology, we categorize the network-based resource allocation problems we study into three main classes: 1) multi-commodity flow problems, 2) multi-commodity flow problems with time-windows and 3) network design problems. In general, these problems are NP-hard problems for which polynomial time solution algorithms do not exist.

For instance, consider a service network design problem, arising in transportation and logistics, subject to resource constraints and variable demands. The objective of the carrier is to determine the cost minimizing or profit maximizing set of services and their schedules. In this problem, several decisions have to be made, such as: pick the best location and size of facilities such that overall costs are minimized; and what is the best fleet mix and size such that service requirements are met? Instances of such problems include: determining the set of flights and their respective schedules for an airline; routing and scheduling tractors and trailers in a trucking operation; and jointly managing the air networks as well as ground networks by determining the routes and schedules for aircraft flights, ground vehicles and package movements for time-sensitive package delivery. All the questions raised here can be formulated as network-based resource allocation problems that fall under the three classes of problems described above.

In this thesis, a *commodity* is defined as a set of packages that needs to be routed on the network, between the same origin and destination (or between the same set of origins and set of destinations). *Multi-commodity flow problems* involve routing multiple commodities over the given network from their origins to destinations, under capacity constraints, in order to minimize costs. If the pickup and delivery of the shipments at their respective origins and destinations are constrained by time windows, this problem is referred to as the *multi-commodity flow problem with time-windows*, and can be harder to solve than simple multi-commodity flow problems. The problem of network design adds another level of complexity to the multi-commodity flow with time-windows problem. Here, the network itself needs to be designed by making decisions regarding the presence or absence of arcs and nodes in the network, and allocating resource supplies, such as capacity, over the network. Moreover, the commodities must be routed optimally on the network to (typically) minimize costs.

In the example of a trucking operation, we find the routes and schedules of tractors, also referred to as vehicles, and trailers, also referred to as shipments. This is an instance of the network design problem. If we add an assumption to the above problem, that the routes of tractors are known, we reduce the scope of the problem. We then need only to route the trailers and schedule these movements. This is an example of a multi-commodity flow problem with time-windows. If we further assume that the tractor movements are scheduled beforehand, and simply route the trailers over the available paths, we are solving a multi-commodity flow problem. Note that in this progression, the scope and complexity of the problem keeps decreasing.

1.3 Problem Description

In this thesis, we shall focus on the class of problems including multi-commodity flow problems with time-windows (*MCFTW*). As mentioned earlier, our goal is to build robust solutions that are less vulnerable to uncertainty. Uncertainty in MCFTW can occur in the form of stochasticity in the supplies and demands of commodities;

available capacities of the network links; and travel and service times on the network. We denote the multi-commodity flow problem with time-windows under uncertainty as *MCFTW-UU.* As illustrated through the examples in this section, the MCFTW is at the core of network design problems, and has sufficient complexity to be of significance in itself. Hence, solving the MCFTW-UU and finding robust solutions is expected to provide insights into the more complex problem of network design under uncertainty.

To illustrate and evaluate our approach, we consider a specific MCFTW(-UU) problem, namely the *Shipment Routing Problem with Time Windows (SRTW)* and the *Shipment Routing Problem with Time Windows Under Uncertainty* (*SRTW-UU*). Under SRTW and SRTW-UU, for each vehicle v in the set of vehicles V, we are given a set of vehicle routes defining a network of locations with time-independent travel times and capacities u_{ij} corresponding to vehicle capacities on the links, and service times at locations. Each shipment k with a single pickup and delivery location and demand d_k needs to be routed over this network, from its origin $O(k)$ to its destination $D(k)$. All units of shipment k must have the same route and schedule. Moreover, shipment k must be picked up after its earliest available time at its origin ($EAT^k_{O(k)}$) and delivered before its latest delivery time at its destination ($LDT^k_{D(k)}$). The objective is to find shipment routes and vehicle and shipment schedules that minimize costs due to vehicle operations, and non-service of shipments. We consider early drop-offs to have no bonuses, and we disallow late drop-offs. We are therefore interested in determining shipment routes and shipment and vehicle schedules, given the *sequence of stops* each vehicle makes. In this work, we are particularly interested in addressing the stochastic nature of input data as seen in penalty costs, vehicle capacities, demands and supplies of shipments, and travel and service times.

SRTW-UU is at the core of the generic network design problem of vehicle routing with pickup and delivery of shipments under time-windows and uncertainty (denoted VRPPDTW-UU [18]). The VRPPDTW-UU reduces to the SRTW-UU if we assume

the routes of vehicles to be known, with the schedule still unspecified. The approach we develop for SRTW-UU problems is directly applicable to MCFTW-UU problems, and provides a natural foundation on which to develop approaches for problems involving network design and resource allocation under uncertainty.

1.4 Challenges

As researchers increasingly recognize the importance of robust solutions that are less susceptible to operational disruption and easier to repair if disrupted, there is a growing body of research and associated literature devoted to the problem of resource allocation under uncertainty. Nonetheless, current capabilities to generate robust solutions to large-scale problems fall short on several dimensions, namely:

1. The robust optimization approaches corresponding to reported successes in the literature are not typically generalizable; instead, they represent approaches highly-tailored to the specific problem being addressed.

2. More general robust optimization approaches, with their underlying non-linear models, do not scale well; resulting in issues for large-scale problems even when solution times are allowed to be long, causing these approaches to be inapplicable when real-time operational solutions are required.

3. Conventional robust optimization models often have associated data requirements that are excessive, requiring data that is either unknown or so extensive as to be too difficult to incorporate into a tractable model. More recent techniques simplify the data requirements, but do not take advantage of information, if known, pertaining to the distributions of pertinent data.

We focus our attention in this thesis on developing an approach that overcomes the above limitations. Designing an approach to create robust plans involves numerous challenges, chief among them are the following:

1. Designing an approach that is tractable for large-scale strategic, tactical and real-time planning problems, when existing techniques are often intractable even for very small-scale problems with no limitations on solution times;

2. Providing guarantees or bounds on the quality of the solution produced; and

3. Developing a robust optimization modeling approach that does not require knowledge of the data distributions underlying the model, but does allow for the capture and exploitation of this data if known. The challenge is to design a model that can incorporate this additional data, but remain tractable. A design goal is that the model's structure should remain the same, whether the data is known, partially known, or unknown.

1.5 Contributions

In addressing the problem of MCFTW-UU, our contributions are as follows:

1. We examine existing approaches of modeling uncertainty and propose suitable modifications, in the context of resource allocation, such that they are applicable to large-scale problems and are more flexible with respect to data requirements.

2. We develop a decomposition scheme that provides a new modeling and algorithmic approach (for MCFTW as well as MCFTW-UU problems) and provides robust solutions that are less vulnerable to uncertainty.

3. We propose a framework for extending the method to address other resource allocation problems, especially those involving network design.

We develop a decomposition approach for MCFTW-UU that:

- provides a new way of modeling the problem by separating the routing and scheduling elements of the problem,

- is flexible with respect to data requirements,

- is applicable to large-scale problems while remaining tractable, and
- is extendable to more complex problems and embeddable within other solution approaches.

1.6 Thesis Structure

In Chapter 2 of this thesis, we address the importance of robustness, and reference related literature. We examine in detail the approaches of Bertsimas and Sim [9], and Charnes and Cooper [12, 13]. We examine these methods in light of the large-scale nature of resource allocation problems, and place particular emphasis on large-scale approaches that are generalizable to a broad class of resource allocation problems.

In Chapter 3, we present various modeling approaches for SRTW-UU, a representative instance of the MCFTW-UU. After describing current approaches, we discuss their shortcomings in addressing the problem, and propose a new *decomposition modeling* framework within which methods from Chapter 2 can be applied to produce robust solutions.

In Chapter 4, we present methods to solve the SRTW-UU (and thus the MCFTW-UU). The algorithm for our decomposition approach is presented and explained with an example. We analyze the quality of the solution generated by our approach with a sample problem, and also examine its performance on a real-world problem. We also explain how this approach is applicable to large-scale problems.

In Chapter 5 we summarize the findings of our work. We discuss the effectiveness of our decomposition approach and the impacts of the robust solutions generated. Finally, we suggest future work in this area.

Chapter 2

Robust and Large-scale Optimization Approaches and Variants

2.1 Introduction to Robust Optimization

Conventionally, problems have been solved assuming the input data to be invariant. However, in practice, the realizations of the input data to the model are, more often than not, different from those assumed in the mathematical model. This causes the solutions that are obtained to be far from optimal in real life, and in some cases, even infeasible.

Models are typically formulated by using '*best-guess*' values or *mean-values* of input data, or by solving '*worst-case*' problems. Such 'worst-case' or 'best-guess' formulations do not give satisfactory solutions. They are either too expensive (worst-case models) or have large errors (mean-value models). We refer to model inputs that are assumed to be realized with certainty, as *nominal* values; the models formulated using nominal inputs as *nominal models*, and the solutions thus obtained as *nominal solutions*. To understand the impact of differences between data realizations and nominal data, *sensitivity analysis* is often employed. However, this is only a post-

optimality study, referred to as *reactive* studies by Mulvey [25], and serves only to identify the extent of the solution's validity. Mulvey also points out the importance of *proactive* approaches to this issue. He argues there is a need for models that "by design, yield solutions that, compared to classical mathematical formulations, are less sensitive to the model data". Mulvey also demonstrates, through the use of examples, how nominal models fall short when uncertainty exists in the real-world realizations of the data. After conducting a case-study on the problems in the Net Lib library [20], Ben-Tal and Nemirovski [6] also concluded that "in the real-world applications of linear programming, one cannot ignore the possibility that a small uncertainty in the data can make the usual optimal solution completely meaningless from a practical viewpoint". In fact, the need for robustness has been recognized in a number of planning and operations areas. Paraskevopoulos et.al. [27] demonstrate this in the case of a capacity planning problem where uncertainty plays an important role.

To adequately address issues of stochastic problem parameters, we need a methodology that produces solutions less vulnerable to uncertainty, allowing us to find solutions that are feasible, and near-optimal, under several data realizations. *Robust optimization*, an approach specifically considering uncertainty, is designed to achieve these goals. In this work, we will refer to models that consider parameter uncertainty and variability in a range of values, and identify those parameter values that are best used in the model, as *robust models*. Solutions produced from such models will be referred to as *robust solutions*.

Researchers have had different notions of robustness, though there is general agreement in the optimization community that a robust solution should reduce the vulnerability of the system. Some researchers define a robust plan as one for which there is a reduced need to re-plan because the plan more frequently remains feasible even as uncertain parameters assume their specific values. Several such metrics exist to measure robustness, with many 'tailored' to the problem under consideration and to reflect its specific vulnerabilities to uncertainty. We will describe metrics we develop

for the MCFTW-UU and SRTW-UU in Chapter 3.

2.1.1 Distinguishing Robust Optimization from Other Methods

It is important to understand that robust optimization differs from existing approaches of handling uncertainty, such as sensitivity analysis and stochastic programming. *Sensitivity analysis* is a post-optimality study that only indicates the extent of the nominal solution's validity. *Stochastic programs*, on the other hand, differ from robust optimization in that they take advantage of the fact that probability distributions governing the data are known or can be estimated. Their goal is to find some policy that is feasible for all (or almost all) possible data instances, while maximizing the expectation of some function of the decisions and the random variables. More generally, such models are formulated, solved analytically or numerically, and analyzed in order to provide useful information to a decision-maker. However this requires an assumption of knowledge of the data distributions which might not always be satisfiable. An important restriction for stochastic programming problems is that the probability distributions of the random parameters are assumed to be known, and cannot depend on the decisions taken.

The most widely applied and studied stochastic programming models are two-stage linear programs, referred to as *stochastic programming with recourse*. Here the decision maker takes some action in the first stage, after which a random event occurs, affecting the outcome of the first-stage decision. A recourse decision can then be made in the second stage, that compensates for any negative effects that might have been experienced as a result of the first-stage decision. The optimal policy from such a model is a single first-stage policy and a collection of recourse decisions (a decision rule) defining which second-stage action should be taken in response to each random outcome.

Dynamic programming is another classical method that allows the incorporation of uncertainty by modeling uncertain parameters as random variables. It can generate provably correct policies that facilitate decision making. However, this method requires knowledge of, or an assumption about the distributions of uncertain parameters, which are not always available, and which moreover, increase complexity of the method greatly. Finally, this method suffers from the well-known *curse of dimensionality* - as problem instances grow in size, the representation of the problem and the resulting solution time grow exponentially.

2.1.2 Robust Optimization Methods

In the linear programming context, the concept of robust optimization is associated with providing *slack* in the solution, as slack allows feasibility of the planned solution for different realizations of the uncertain data. Instead of finding an optimal solution to a single instance of the problem using nominal values of the data, the goal is to find a solution that remains near-feasible and near-optimal when the data changes.

An early step in this direction was taken by Soyster [31], who proposes a worst-case linear optimization model in which each data element takes on its *extreme*, or worst-case, value that is at the boundary of its range of values. That is, for each uncertain parameter, the model computes a solution assuming that the realization of the parameter that would consume the maximum possible resources of the system. Such a solution would, in fact, have the highest possible slack in the system for any realization of the uncertain parameters. However, Ben-Tal and Nemirovski [7] argue that such solutions are too conservative because low probability events force a very costly solution relative to that achieved for the nominal problem.

Bertsimas and Sim [9] note that the probability of all the data points simultaneously taking on their worst case values, as in Soyster's model, is very low. Moreover, it is not practical to plan for maximum uncertainty, as robustness comes with an associated cost. Instead, they argue that decisions are to be made which attain a reasonable

trade-off between the degree of robustness and cost spent. In 2002, Bertsimas and Sim proposed a linear model in which some of the data elements take on extreme or worst-case values, while the others assume their nominal values. The degree of robustness can be controlled by adjusting a *level of robustness* parameter Γ, which guarantees a feasible solution to any instance in which fewer than Γ coefficients realize their worst-case values. The resulting robust counterpart to the linear program is still linear, and has an intuitive appeal in that the tradeoff between robustness and solution cost can be computed by varying Γ. Moreover the data requirements of this model are low, matching the availability (or lack thereof) of data for many real-world problems. This approach is of particular interest to us, and we shall examine it in more detail in Section 2.4.

In another approach addressing the issue of over-conservatism, Ben-Tal and Nemirovski [6, 7] propose the usage of ellipsoidal uncertainties, which result in less conservative solutions than that of Soyster's method. The principal motivation for using this kind of uncertainty set is that measurement errors are typically distributed in an ellipsoid centered at the mean of the distribution. Under this uncertainty assumption, linear programs have robust counterparts that are conic quadratic problems. However, an important drawback of this approach is that the resulting models, though convex, are non-linear and computationally more difficult than Soyster's models, especially under integer constraints and for large-scale problems.

Yet another approach to robust optimization, called chance-constrained programming, was introduced by Charnes and Cooper in 1959 [12]. This work was among the very first to address the problem of planning under uncertainty. Chance-constrained programming (CCP) admits random data variations and permits constraint violations up to certain specified probability limits [13]. It is not required (unlike stochastic programming) that the decisions be feasible for (almost) every outcome of the random parameters, but require feasibility with at least some specified probability. When uncertainty in a constraint is limited to the right-hand-side, Charnes and Cooper show

that the randomness component can be modeled deterministically and the model can be converted into a linear program with very little increase in complexity. The model is intuitively appealing, but requires at least partial knowledge of data distributions. CCP models, however, face computational issues with the addition of uncertain parameters in a constraint. Under uncertainty in the left-hand-side of the constraint matrix, the usage of multinomial distributions becomes necessary, and serious computational difficulties are encountered. Most chance-constrained programming applications have been limited to capturing uncertainty only in the right-hand side due to these computational difficulties. Chance-constrained programming has been applied to, among other areas, critical path analysis [14], networks [16] and research and development projects [17].

In addition, several problem specific approaches to protect solutions against uncertainty have been explored. Paraskevopoulos et.al. [27] solve a capacity planning problem, using a *sensitivity approach*, by minimizing an augmented objective function that penalizes the sensitivity of the objective function to various types of uncertainty. They cast the problem in a deterministic framework in order to avoid complications inherent to nonlinear stochastic formulations, thus maintaining computational simplicity. Lan, Clarke and Barnhart [23] propose an '*intelligent routing model*' for aircraft routing to reduce propagation of delays along the downstream flight legs. In addition, they introduce a new approach to minimize the number of passenger misconnections by re-timing the departure times of flight legs within a small time window. Their approach helps to improve passenger connection times without significantly increasing costs.

Kang and Clarke [22] present an approach to improve operational robustness and improve airline quality using *degradable airline scheduling*, an approach to derive robust airline schedules that simplify recovery operations. Prioritizing on the basis of revenue, schedules are divided into layers that provide airlines with a clear delay/cancellation policy and might enable them to market and sell tickets for flight

legs based on passenger preference for reliability. Different models of incorporating degradability are presented, namely the degradable schedule partitioning model, the degradable fleet assignment model, and the degradable aircraft routing model.

Shebalov and Klabjan [30] propose robust approaches tailored to specific instances of crew scheduling problems, which are instances of the MCFTW-UU with the schedule defined apriori, by exploiting the specialized structure of the problem. The authors introduce the concept of *move-up crews* and improve costs by swapping crews, and show the resulting benefits.

The methods presented in the literature thus far can be divided into three broad categories: problem specific (e.g. Shebalov and Klabjan, Lan et. al., Kang and Clarke, Paraskevopoulos et. al.), distribution free and general (e.g. Soyster, Bertsimas-Sim) and those that are general but use knowledge of uncertainty distributions (e.g. CCP).

Among the methods discussed above, the Bertsimas-Sim and Chance-constrained programming approaches allow for a controlled tradeoff of robustness with cost. Moreover, they are applicable, in theory, to all linear or integer programs. In both these approaches, the 'robust' linear program can be converted into a deterministic program that is still linear. Hence these methods are computationally attractive, especially for large-scale problems. We detail these methods, which are of special interest to us, beginning in Section 2.4 onwards.

We shall devote the next two sections to discussing the implications of large-scale problem structure on the solution of network-based resource allocation problems under uncertainty.

2.2 Large-scale approaches

As discussed in Chapter 1, network-based resource allocation problems are often large-scale, and often require integer solutions. As such, specialized methods are re-

quired to solve them. Problems requiring binary integer solutions can be solved using integer programming solution approaches, such as *branch-and-bound.* Branch-and-bound is an efficient solution enumeration technique that finds an optimal solution by repeatedly solving the linear programming (LP) relaxation of the integer problem with the binary constraints relaxed to allow variable values between (or at the boundaries) of 0 and 1 [3]. A branch-and-bound 'tree' is constructed, with a linear programming relaxation typically solved at each node of the tree. Given the LP solution, branching decisions set fractional variable values to 0 or 1, thereby defining a new LP. The process repeats until an integer solution (if it exists) is found. The best integer solution is recorded, and continually updated, as new solutions are found. The value of the best solution is used to identify nodes (or LP problems) in the branch-and-bound tree that cannot lead to improved solutions. These LPs, and others, need not be solved and hence, optimal solutions can be identified without evaluating all possible solutions.

In the case of very large-scale problems, it can be impractical to solve even the linear programming (LP) formulation directly. Instead, an indirect technique called *column generation* is employed. Here the word 'column' indicates 'variable'. The approach is to enumerate only some, not all, variables, generating them on an as-needed basis. The optimal solution, according to linear programming theory, uses only a subset of the total number of variables; hence, the goal is to identify those variables that are present in the optimal solution. The challenge here is to identify the subset of relevant variables as efficiently as possible [2]. Column generation consists of the following steps [3]:

- Step 1: The *Restricted Master Problem* (RMP), a linear program containing only a subset of the variables is constructed and solved.

- Step 2: Generate one or more variables that might improve the LP solution by solving the *Pricing Problem.* If no such variables are generated, STOP; the LP problem is solved.

- Step 3: Add the variables generated in the pricing problem to the RMP and return to Step 1.

By appropriately defining the pricing problem, the column generation approach is guaranteed to find, for any LP, an optimal solution. Solving large-scale binary integer programs, however, involves a combination of both column generation and branch-and bound techniques. This combined methodology is referred to as *branch-and-price*, in which column generation is used to solve each LP solved in the branch-and-bound tree. Critical to the success of branch-and-price solution approaches are:

1. a tractable solution algorithm for the pricing problem; and
2. branching strategies in the branch-and-price algorithm that allow the same algorithm to be used to solve the pricing problem for each LP solved in the branch-and-bound tree, while ensuring that all branching decisions are enforced in the solution.

Further details of the challenges associated with branch-and-price solution approaches are provided in Barnhart et. al. [4] and Desrosiers et al. [19].

An approach analogous to column generation is *row generation*, where, instead of columns, rows or constraints are generated on an as-needed basis. This is used in the case when there are a very large number of constraints or a set of 'hard' constraints that make the problem difficult to solve. 'Hard' constraints are omitted from the constraint set, and a *relaxed* master problem with a smaller set of constraints is first solved. The resulting solution is examined for feasibility with respect to the omitted constraints. If these constraints are also satisfied, then the solution is optimal, otherwise selected constraints are added back into the master problem to obtain a better solution. This process continues until a feasible, and thus optimal, solution is found.

Typically, resource allocation problems of large size require application of not only one, but a combination of the aforementioned approaches. These are called *branch-*

and-price-and-cut algorithms, which combine column as well as row generation within the framework of branch-and-bound. This indicates that the robust approaches that we will examine should also have formulations that allow the efficient utilization of such large-scale approaches.

2.3 Robust, Large-scale Approaches

Mulvey[25] tailors his robustness methods to large-scale problems using a scenario-based approach. His algorithm can be extended to large-scale problems by taking advantage of high-performance computers and advances such as parallel computing. However, because the approach requires the enumeration of all scenarios against which the solution is to be protected, computational tractability can quickly become an issue as problem size or the number of scenarios grow.

Among the robust approaches discussed in Section 2.1.2, not all can be applied to large-scale problems because the robust solution methodology is not compatible with existing large-scale approaches. Hence there is a need to adapt robustness approaches to existing large-scale approaches, or to develop novel large-scale, robust optimization approaches.

In the rest of this chapter, we focus on the Bertimas-Sim approach and the Chance Constrained Programming (CCP) approach, with specific emphasis on applying them to large problems.

2.4 Robust Formulation of Bertsimas and Sim

Consider a standard linear program, that is:

$$\max \quad \mathbf{c}^T\mathbf{x} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j\in J} a_{ij}x_j \leq \mathbf{b} \tag{2.2}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \tag{2.3}$$

Soyster [31] considers column-wise uncertainty, where each column A_j of the constraint matrix belongs to a convex set K_j. He shows that the above problem is equivalent to the following robust formulation:

$$\max \quad \mathbf{c}^T\mathbf{x} \tag{2.4}$$

$$\text{s.t.} \quad \sum_{j\in J} \bar{a}_{ij}x_j \leq \mathbf{b} \tag{2.5}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \tag{2.6}$$

where $\bar{a}_{ij} = \sup_{A_j\in K_j}(a_{ij})$. This means that extreme (or worst-case) values of coefficients that effectively maximize the amount of slack for the nominal problem are used in the 'robust' model. The use of worst-case values results in solutions that are far from optimal for many realizations of the constraint matrix coefficients.

Bertsimas and Sim [9] argue that worst-case approaches such as that of Soyster, are too conservative, and hence, expensive. Instead, they suggest an approach aimed at avoiding the overly conservative tendencies of Soyster's approach by providing a mechanism to control the 'degree of conservatism'.

In the approach of Bertsimas and Sim, all uncertainty is assumed to be located in the coefficients of the $\mathbf{A}$ matrix. By performing some simple transformations and rewriting $\mathbf{A}$, uncertainty in $\mathbf{c}$ and $\mathbf{b}$ can also be captured. By changing the objective function to maximize z and adding the constraint $z - \mathbf{c}^T\mathbf{x} \leq 0$, the objective function can be moved into the $\mathbf{A}$ matrix, thus enabling uncertainty in the objective function

coefficients to be captured. Similarly, if we have uncertainty in the right-hand-side **b**-vector, the **b**-vector values can be subtracted from the left-hand side and the right-hand side can be replaced by zero. The assumption of uncertainty in the **A**-matrix therefore incurs no loss of generality.

Each entry of of the left-hand side of the constraint matrix, **A**, is assumed to be a random variable with $\tilde{a}_{ij}$ being the symmetric, unbounded variable corresponding to the (i,j)th entry of **A**. No actual probability distribution of the random variable is assumed, only an interval of values that $\tilde{a}_{ij}$ can assume. Specifically, a_{ij} denotes the nominal value of $\tilde{a}_{ij}$, which is used in the deterministic formulation, and $\hat{a}_{ij}$ is the half-interval of $\tilde{a}_{ij}$. Hence, $\tilde{a}_{ij}$ can take on values in the interval $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$ and the nominal value a_{ij} is the mean value of the symmetric distribution. The *extreme values* that $\tilde{a}_{ij}$ can take are $a_{ij} - \hat{a}_{ij}$ and $a_{ij} + \hat{a}_{ij}$.

Let J_i be the set of coefficients for constraint i that are subject to parameter uncertainty, that is, $\tilde{a}_{ij}, j \in J_i$ takes values from a symmetric distribution as described above. For each constraint i, there is a parameter Γ_i which can take a (possibly continuous) value in the interval $[0, |\mathbf{J}_i|]$. Because it is unlikely that all $|J_i|$ coefficients will assume their worst-case (or extreme) values, Γ_i is used as a means of adjusting the 'level of protection'. The Bertsimas-Sim formulation protects against the case when up to γ_i of the $|J_i|$ coefficients are allowed to assume their extreme values, for all constraints i.

The corresponding robust non-linear model according to the Bertsimas-Sim model can then be written as:

$$\max \quad \mathbf{c}^T\mathbf{x} \tag{2.7}$$

$$\text{s.t.} \sum_j a_{ij}x_j + \max_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \notin S_i\}} \{\sum_{j \in S_i} \hat{a}_{ij}y_j + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i}y_t\} \leq b_i \quad \forall i \tag{2.8}$$

$$-y_j < x_j < y_j \quad \forall j \tag{2.9}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \quad \forall j \tag{2.10}$$

$$y \geq 0 \tag{2.11}$$

Because Γ_i can take on continuous values, up to $\lfloor \gamma_i \rfloor$ of the coefficients $\tilde{a}_{ij}$ in constraint i are allowed to take on their worst-case values, and one coefficient a_{it} changes by $(\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it}$. In the above formulation, S_i represents the set of uncertain parameters in constraint i that take on their extreme values, such that $|S_i| = \lfloor \Gamma_i \rfloor$, $S_i \subseteq J_i$. $\{t_i\}$ indicates the coefficient a_{it_i}, for constraint i, that changes by $(\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i}$.

For the ith constraint, the term $\max_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| = \lfloor T_i \rfloor, t_i \in J_i \notin S_i\}} \{\sum_{j \in S_i} \hat{a}_{ij}y_j + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i}y_t\}$ is a *protection function* that protects against the worst-case realizations of all $\tilde{a}_{ij}, j \in J_i$. The parameterized protection function thus uses Γ_i to offer various *levels of protection.* $\lfloor \Gamma_i \rfloor$ indicates the minimum number of coefficients in constraint i that can assume their worst case values without destroying feasibility of the solution. $\Gamma_i = 0$ represents the deterministic or nominal case, whereas $\Gamma_i = |J_i|$ reduces this formulation to the Soyster formulation.

Bertsimas and Sim [9] prove that the above non-linear formulation (2.7) - (2.11) can be cast as a deterministic linear program, as follows:

$$\max \mathbf{c}^T\mathbf{x} \tag{2.12}$$

$$\text{s.t.} \sum_{j\in J} a_{ij}x_j + z_i\Gamma_i + \sum_{j\in J_i} p_{ij} \leq b_i \qquad \forall i \in I \tag{2.13}$$

$$z_i + p_{ij} \geq \hat{a}_{ij}y_j \qquad \forall i \in I, \forall j \in J_i \tag{2.14}$$

$$-y_j \leq x_j \leq y_j \qquad \forall j \in J \tag{2.15}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{2.16}$$

$$p_{ij} \geq 0 \qquad \forall i \in I, \forall j \in J_i \tag{2.17}$$

$$y_j \geq 0 \tag{2.18}$$

$$z_i \geq 0 \tag{2.19}$$

The detailed proof of the equivalence of (2.12)-(2.19) with (2.7)-(2.11) is in [9].

Thus, the Bertsimas-Sim robust optimization approach ensures that the form of the math program remains linear, and hence more tractable than formulations with non-linearities. Bertsimas and Sim [9] also provide probabilistic guarantees on the feasibility of constraints when more than Γ_i coefficients take on their worst-case values. Moreover, they show how this formulation can be applied to portfolio optimization, knapsack problems, supply chain management [10], and network flows [8] in order to obtain robust solutions.

The **advantages** of the Bertsimas-Sim model are:

- It is generally applicable to linear programs and integer programs.
- Linear integer programs remain linear integer programs, but contain more variables, degrading tractability minimally.
- Probability distributions for the uncertain data are not required to be known. Uncertainty can be captured knowing the symmetric bounds of variation alone.

- It allows adjustments to the 'level of robustness' using the Γ parameter, thereby providing measures of the *price of robustness*, that is, the changes in planned objective function value with changes in protection level. Robustness involves backing off from optimality to gain solutions less vulnerable to uncertainty, implying that there is a price associated with achieving each level of robustness.

- This model, with minor alterations, can also capture simple correlations between uncertain data in a constraint [9]. However, it cannot capture correlations among uncertain data across constraints.

The approach, however, also has some **limitations**:

- To determine the change in planned costs (or profits) as a function of the level of 'protection', the problem has to be re-solved multiple times, once for each different value of Γ_i, for all i. Because the bounds are also not tight, there are very few guidelines to the choice of Γ_i. This poses computational challenges for large-scale problems.

- It assumes symmetric and bounded distributions of uncertainty of parameters about their nominal values.

- It does not incorporate knowledge of probability distributions, if known. This can result in lack of inclusion of problem knowledge in the model.

- Probability bounds of constraint violation are derived for each constraint, and cannot be easily extended to an overall protection level for the system.

- As discussed in section 2.5, this approach is not particularly well-designed for the solution of very large-scale resource allocation problems.

Further, recent investigations by Sakamoto [28] and Bryant [11] report that the Bertsimas-Sim approach works better in cases where the error is small, rather than where the uncertainty ranges are large. Moreover, they report issues in applying this approach to discrete optimization problems, especially binary integer programs, and

for large-scale problems. Results from these works are described in greater detail in Section 2.8.

2.5 Extended Bertsimas-Sim (Delta) Formulation

Taking into account the above limitations of the Bertsimas-Sim formulation as applied to resource allocation problems, we propose an alternative *Delta* formulation which preserves the spirit of the basic formulation while addressing some issues. This alternative formulation is designed for binary integer programs, the type of problems we address in this thesis.

The standard binary integer program that is required to be made robust is:

$$\max \quad \mathbf{c}^T\mathbf{x} \tag{2.20}$$

$$\text{s.t.} \quad \sum_{j\in J} a_{ij}x_j \leq b_i \quad \forall\, i \in I \tag{2.21}$$

$$\mathbf{x} \in \{0,1\} \tag{2.22}$$

The basic practical issue to be addressed is to select the appropriate level of protection Γ_i for each constraint i. A planner must specify a level of protection for *each* constraint or type of constraint; a potentially cumbersome task for large-scale problems. Given this, it might be necessary to solve the Bertsimas-Sim model repeatedly for varying values of the Γ_i parameters before a satisfactory solution is identified. Because solving the model even once can be computationally challenging, the requirement to solve it multiple times is likely to be impracticable for large problems.

To avoid the need to specify Γ values, we modify the Bertsimas-Sim formulation to include a constraint requiring the total profit of the robust solution to be within a difference of δ from the nominal optimal value. Additionally we change the objective to one of minimizing the maximum number of variables that *must* assume their nominal, rather than extreme, values to satisfy all constraints. We refer to this model as

the Delta formulation, derived from the added constraint on profit.

We define *variable* Δ_i equal to the maximum number of variables x in the solution with $x = 1$ whose coefficient values must assume their nominal values for constraint i to remain feasible. The objective function value, denoted ν, equals the maximum value of Δ_i over all constraints i. We arrange, for each constraint i, its associated binary decision variables, x_j, in increasing order of their $\hat{a}_{ij}$ values. After ordering, the position of the jth column in the ith row is denoted by $l(i,j)$. For example, the variable j in constraint i with the smallest $\hat{a}_{ij}$ value has $l(i,j) = 1$. Also the variable with the largest $\hat{a}_{ij}$ value has $l(i,j) = N$, with N equal to the number of binary variables.

Data:

- c_j: profit coefficient for variable j.
- I: set of constraints.
- J: set of variables.
- $\tilde{a}_{ij}$: realized constraint coefficient for constraint i and variable j, $\forall\, i \in I,\, j \in J$.
- a_{ij}: Nominal value of $\tilde{a}_{ij}$, also the mean value of its symmetric range of variation, $\forall\, i \in I,\, j \in J$.
- $\hat{a}_{ij}$: Half-interval of symmetric range of variation of $\tilde{a}_{ij}, \forall\, i \in I, j \in J$.
- b_i: Right-hand side value for ith constraint, $\forall\, i \in I$.
- N: number of variables.
- $l(i,j)$ = the ranking of the jth variable when the $\hat{a}_{ij}$ values in constraint i are sorted in increasing order.
- $j(i,l)$: is the original index (j) of the variable that takes the lth position in the sorted $\hat{a}_{ij}$ values for constraint i.

- y_j^*: optimal value of variable j for the nominal problem, $\forall\, j \in J$.

- δ: user-specified incremental cost that is acceptable for increased robustness, that is, the profit of a robust solution from the Delta formulation is at least $\sum_{j \in J} c_j y_j^* - \delta$.

Decision variables:

- x_j: binary decision variable that equals 1 if variable is present in the solution and 0 otherwise.

- s_{ij}: equals 1 if the coefficient $\tilde{a}_{ij}$ is not allowed to take on its extreme value, and takes on its nominal value in the solution.

- Δ_i: the maximum number of variables x in the solution (with $x = 1$) whose coefficient values must assume nominal values for constraint i to remain feasible

- ν: the maximum number of uncertain coefficients in any constraint that must assume nominal values, rather than their extreme values, to satisfy the constraints

This leads to the following Delta formulation:

$$\min \quad \nu \tag{2.23}$$

$$\text{s.t.} \quad \sum_j c_j x_j \geq \sum_j c_j y_j^* - \delta \tag{2.24}$$

$$\nu \geq \Delta_i \qquad \forall\, i \in I \tag{2.25}$$

$$\sum_j (a_{ij} + \hat{a}_{ij}) x_j - \sum_j \hat{a}_{ij} s_{ij} \leq b_i \quad \forall\, i \in I \tag{2.26}$$

$$s_{ij} \leq x_j \qquad \forall\, i \in I, j \in J \tag{2.27}$$

$$\Delta_i \geq l s_{ij(i,l)} - \sum_{k=1}^{l} [1 - x_{j(i,k)}] \quad \forall\, l = 1 \ldots J, \forall\, i \in I \tag{2.28}$$

$$x_j \in \{0, 1\} \qquad \forall\, i \in I, j \in J \tag{2.29}$$

$$s_{ij} \in \{0, 1\} \qquad \forall\, i \in I, j \in J \tag{2.30}$$

The formulation is described as follows: The objective is to minimize the maximum number of coefficients that must assume their nominal values to satisfy all constraints. This serves the purpose of trying to maximize the number of coefficients that can take their extreme values and still maintain feasibility. Constraints (2.24) require that the profit from the 'robust' solution not differ from the profit of the deterministic solution by more than a 'robustness budget' of δ. Requirements (2.25) ensure for each constraint i that ν is greater than or equal to Δ_i, that is, the maximum number of coefficients in constraint i that must assume nominal values to maintain feasibility. Because the objective is to minimize, ν exactly equals the maximum value of Δ_i, over all constraints i. Feasibility is assured by constraints (2.26). We set s_{ij} equal to 1 in constraint i, for all coefficients j that must be set to their nominal values. Inequalities (2.27) set s_{ij} to zero if x_j is zero in the solution. Constraints (2.28) provide a mechanism to count the maximum number of variables in constraint $i \in I$ whose coefficients must take on nominal values. The explanation for this constraint lies in the realization that when the columns are sorted in increasing order of $\hat{a}_{ij}$ values for each row i, the *maximum number* of coefficients that must assume nominal values to maintain feasibility is determined by forcing the smallest $\hat{a}_{ij}$s in the solution to have their associated s_{ij} values set to 1, if x_j is in the solution. The x and s variables are binary, as required by (2.29) and (2.30) respectively. Alternatively, one can think of this model as maximizing the minimum number of coefficients in a constraint that can take on their worst-case values, under budget limitations.

The mechanism to find the maximum number of nominal-valued coefficients in a constraint is best explained using an example. Consider a constraint i with $|J| = 5$, with $\hat{a}_{ij}$ values of 7, 1, 2, 0.5, and 10. Arranging these in ascending order, with indices denoted l, we have 0.5, 1, 2, 7 and 10. Let $x^*_{j(i,1)}$ be zero, and $x^* = 1$ for all other j. Constraints (2.27) set $s_{i,j(i,1)} = 0$ and allow $s_{i,j(i,l)}$ to equal 0 or 1 for all other l. Say, for this solution, constraint i is violated by 4 if the coefficients for all variables in the solution are set to their extreme values, that is, $\sum_{j \in J}(a_{ij} + \hat{a}_{ij})x_j \leq b_i$ is violated by 4, with $b_i = 6$. To achieve feasibility of constraint (2.26), possible solutions are to set:

1. $s_{i,j(i,2)} = s_{i,j(i,3)} = s_{i,j(i,4)} = 1$ and $s_{i,j(i,1)} = s_{i,j(i,5)} = 0$;

2. $s_{i,j(i,3)} = s_{i,j(i,4)} = 1$ and $s_{i,j(i,1)} = s_{i,j(i,2)} = s_{i,j(i,5)} = 0$;

3. $s_{i,j(i,4)} = 1$ and $s_{i,j(i,1)} = s_{i,j(i,2)} = s_{i,j(i,3)} = s_{i,j(i,5)} = 0$;

4. $s_{i,j(i,5)} = 1$ and $s_{i,j(i,1)} = s_{i,j(i,2)} = s_{i,j(i,3)} = s_{i,j(i,4)} = 0$;

5. $s_{i,j(i,4)} = s_{i,j(i,5)} = 1$ and $s_{i,j(i,1)} = s_{i,j(i,2)} = s_{i,j(i,3)} = 0$.

Note that only the first solution is allowed, as it represents the solution with the maximum number of coefficients that *must* assume their nominal values to maintain feasibility of the constraint. Note further that Δ_i, as defined by constraints (2.28), equals 3 for solution 1; equals 3 for solution 2; equals 3 for solution 3; equals 4 for solution 4; and equals 4 for solution 5. Constraints (2.28) ensure that the invalid solutions represented by 2), 3) and 4) do not define the value of the maximum number of coefficients by inflating the number of variables in each of these solutions with s-values set to 1 to at least the value of the true maximum. Because there exists another solution with the same x-values and different s-values whose objective function value 2.23 is improved (and is correctly computed), a solution with the correct Δ_i, for all i, will always be selected.

Observe further that constraints of the form (2.27) can be violated only when $x_j = 0$ for some j and for some i in a solution. Because constraints (2.26) are 'less than or equal to' constraints, they will not be violated when $x_j = 0$ and $s_{ij} = 1$. However, because the objective is to minimize, the largest Δ_i will be required to be as small as it possibly can, which happens when $s_{ij} = 0 \forall i \in I$ when $x_j = 0$. Therefore, constraints (2.27) will never be violated in an optimal solution, and may be relaxed.

2.5.1 Investigation of Large-scale Extension Capabilities

For the Delta formulation to be practical for use in solving large-scale resource allocation problems, it should be amenable to column generation techniques, as explained

in Section 2.2. We check this condition for a simple variant of the MCFTW-UU problem, namely the multi-commodity flow problem under uncertainty. If the path-based multi-commodity flow problem under uncertainty recast using the Delta model allows efficient column generation, we conclude that the Delta model is applicable to large-scale problems.

In this context, efficient column generation is possible if the maximum reduced cost variable (in this case, a path) can be identified without enumerating all the variables. Specifically, if the reduced costs on paths can be expressed in terms of arc costs, shortest path algorithms can be used to identify the most positive reduced cost variable and efficient column generation is possible.

The following is the integer multi-commodity flow formulation.

Data:

- c_{ij}: Profit on arc (i, j)
- c_p: Profit due to 1 unit of flow on path p, equivalent to the sum of profits of the arcs on the path. $c_p = \sum_{(i,j) \in p} c_{ij}$
- A: Set of arcs in the network
- K: Set of commodities to be transferred from their respective origins to their respective destinations
- d_k: Required flow of commodity k between the demand and supply nodes of commodity k.
- δ_{ij}^p: Arc-path indicator variable that is equal to 1 if arc (i, j) is on path p, 0 otherwise
- u_{ij}: Capacity of arc (i, j)
- P^k: The set of origin to destination paths for commodity k

- N: The total number of paths in the network. $N = \sum_{k \in K} |P^k|$

Variables:

- f_p: equals 1 if all d_k units of commodity k flow on any path $p \in P_k$; and equals 0 otherwise.

$$\max \quad \sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p \tag{2.31}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{p \in P^k} d_k \delta_{ij}^p f_p \leq u_{ij} \quad \forall\, (i,j) \in A \tag{2.32}$$

$$\sum_{p \in P^k} f_p = 1 \qquad \forall\, k \in K \tag{2.33}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K \tag{2.34}$$

The objective in the above formulation is to maximize profits due to commodity flows on the network. The constraints correspond to choosing flows satisfying capacity constraints and satisfaction of demands on the network.

The Delta version of formulation (4.1)-(4.4) considering uncertainty in demands, is as follows. We define the following additional notation:

- $f_p^{'*}$: is the optimal solution to (4.1)-(4.4) using nominal values of data;
- δ: is the user-specified incremental cost that is acceptable for increased robustness, that is, the profit of a robust solution from the Delta formulation is at least $\sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p^{'*} - \delta$;
- $l(ij,p)$: the ranking of the pth variable when the $\hat{d}_k$ values in each capacity constraint of form (4.2) corresponding to arc (i,j) are sorted in increasing order;
- $p(ij,l)$: is the original index (p) of the variable that takes the lth position in the sorted $\hat{d}_k$ values of the capacity constraint corresponding to arc (i,j);

- ν: is the maximum number of uncertain coefficients that must assume their nominal values, rather than their extreme values, to satisfy all constraints;

- Δ_{ij}: is the maximum number of variables f in the solution (with $f = 1$) whose coefficient values must assume their nominal values for the capacity constraint for arc (i, j) to remain feasible; and

- s_{ij}^p: equals 1 if the coefficient $\tilde{d}_k$ (with $\delta_{ij}^p = 1, p \in P^k$) is not allowed to take on its extreme value in the capacity constraint for arc (i, j), and takes on its nominal value in the solution to maintain feasibility.

The duals associated with the constraints containing the variable f_p are shown in the right-most column. γ, π, μ and α are non-negative variables and σ are unrestricted in sign.

$$\min \nu \tag{2.35}$$

$$\text{s.t.} \sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p \geq (\sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p)^* - \delta \qquad \gamma \tag{2.36}$$

$$\nu \geq \Delta_{ij} \qquad \forall (i,j) \in A \tag{2.37}$$

$$\sum_{k \in K} \sum_{p \in P^k} (d_k + \hat{d}_k) \delta_{ij}^p f_p - \sum_{k \in K} \sum_{p \in P^k} \hat{d}_k \delta_{ij}^p s_p^{ij} \leq u_{ij} \qquad \forall (i,j) \in A \qquad -\pi_{ij} \tag{2.38}$$

$$\Delta_{ij} \geq l s_{p(ij,l)}^{ij} - \sum_{m=1}^{l} [1 - f_{p(ij,m)}] \qquad \forall\, l = 1 \ldots N, \forall (i,j) \in A \qquad -\mu_{ij}^l \tag{2.39}$$

$$\sum_{p \in P^k} f_p = 1 \qquad \forall\, k \in K \qquad \sigma_k \tag{2.40}$$

$$f_p \geq s_p^{ij} \qquad \forall (i,j) \in A, \forall\, p \in P^k, \forall\, k \in K \qquad \alpha_{ij}^p \tag{2.41}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K \tag{2.42}$$

$$s_p^{ij} \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K \tag{2.43}$$

In Section 2.5 we show that constraints of the type (2.41) be relaxed. Therefore, the reduced cost of variable f_p

$$= \quad 0 - \{\gamma d_k c_p - \sum_{(i,j)\in A} \pi_{ij}(d_k + \hat{d}_k)\delta_{ij}^p - \sum_{(i,j)\in A} \sum_{l=1}^{N} \mu_{ij}^{l(ij,p)} + \sigma_k\} \tag{2.44}$$

$$= \quad \sum_{(i,j)\in A} \left\{ \delta_{ij}^p \{-\gamma d_k c_{ij} + \pi_{ij}(d_k + \hat{d}_k)\} + \sum_{l=1}^{N} \mu_{ij}^{l(ij,p)} \right\} - \sigma_k. \tag{2.45}$$

This expression for reduced cost is similar to that of standard multi-commodity flow formulations. This indicates that negative reduced cost variables, even though not present in the current formulation, may be identified by solving shortest path algorithms with modified costs on the network arcs. If a path with negative reduced cost exists, it indicates a variable with negative reduced cost, which is added to the formulation, and the resulting LP re-solved. If no such path exists, the current solution is optimal for the linear programming relaxation. Because the Delta formulation allows us to price-out variables that are not currently present in the formulation, it is amenable to column generation approaches.

2.6 Chance-Constrained Programming (CCP)

Charnes and Cooper [12] were among the first to address the problem of robust planning under uncertainty. Because uncertain input data can lead to constraint violations once a plan is put into operation, they regulate the chance that any constraint is violated, hence the name "chance-constrained programming" (CCP).

Charnes and Cooper define chance-constrained programming to admit random input data variations and permit constraint violations up to specified probability limits [12]. CCP thus requires that each of the constraints of the LP be satisfied with some user-specified probability.

To illustrate the basic principle of CCP, consider the following basic LP:

$$\max \quad \mathbf{c}^T\mathbf{x} \tag{2.46}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{2.47}$$

$$\mathbf{x} \geq 0. \tag{2.48}$$

When $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ are random variables with known distributions, we can write a chance-constrained model in which the probability of constraint feasibility is represented by a vector of constants $\boldsymbol{\alpha}$. The idea is that given certain constraints with uncertain data, we want the probability of the constraints being satisfied to be greater than $\boldsymbol{\alpha}$. Thus the chance-constrained model for (2.46)-(2.48) is of the form:

$$\text{optimize} \quad f(\mathbf{c}, \mathbf{x}) \tag{2.49}$$

$$\text{s.t.} \quad P(\mathbf{A}\mathbf{x} \leq \mathbf{b}) \geq \boldsymbol{\alpha} \tag{2.50}$$

$$\mathbf{x} \geq 0. \tag{2.51}$$

where 'P' means 'probability'. In the vector $\boldsymbol{\alpha}$, each element α_i $(0 \leq \alpha_i \leq 1)$ is a constant, referred to as the *protection level* for constraint i, and (1 - α_i) specifies the maximum degree of violation of constraint i. Thus, the ith constraint $\sum_{j=1}^{N} a_{ij}x_j \leq b_i$ is converted to $P(\sum_{j=1}^{N} a_{ij}x_j \leq b_i) \geq \alpha_i$ in the CCP formulation, which means that the ith constraint is allowed to be violated at most $(1 - \alpha_i)$ proportion of the time.

Charnes and Cooper translate (2.49)-(2.51) into different models, for varied types of objective functions and correspondingly different constraints. They present models for three types of objective functions, namely: the expected value optimization model (E-model), the minimum variance (or mean-square error) objective (V-model) and a maximum probability model (P-model). In each of these models, $\mathbf{b}$ and $\mathbf{c}$ are assumed to be uncertain. Further details of these models are given in [13] and [15].

Further, Ben-Israel [5] shows that

$$P(\sum_{j=1}^{N} a_{ij}x_j \leq b_i) \geq \alpha_i \Leftrightarrow \sum_{j=1}^{N} a_{ij}x_j \leq F_{b_i}^{-1}(1-\alpha_i), \tag{2.52}$$

with $y = F_{b_i}^{-1}(1-\alpha_i)$ equal to the *quantile* value in the cumulative distribution function (CDF), F_{b_i}, of b_i such that the probability that b_i takes on values less than or equal to y is (1- $\boldsymbol{\alpha}$). That is, if $f(b_i)$ is the probability distribution function of b_i, $\int_{-\infty}^{y} f(b_i)db_i = 1-\alpha_i$.

The expression on the right of (2.52) is a linear programming constraint. Suppose the objective function in (2.49)-(2.51), that is, optimize $f(\mathbf{c},\mathbf{x})$, assumes the form $\max E(\mathbf{c}^T\mathbf{x})$, where $\mathbf{c}$ is a vector of random variables. Let $E(\mathbf{c}^T\mathbf{x}) = \boldsymbol{\mu}_{\mathbf{c}}^T\mathbf{x}$ so that $\boldsymbol{\mu}_{\mathbf{c}}$ is a vector whose elements are the expected values (means) of the elements in $\mathbf{c}$.

The linear program is then

$$\begin{aligned} \max \quad & \boldsymbol{\mu}^T\mathbf{x} && (2.53)\\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq F_{\mathbf{b}}^{-1}(1-\boldsymbol{\alpha}) && (2.54)\\ & \mathbf{x} \geq 0. && (2.55) \end{aligned}$$

From the distributions of the elements in $\mathbf{c}$, we can determine a vector of stipulations $\boldsymbol{\beta}$ such that $P(\mathbf{c} \leq \boldsymbol{\lambda}_c) \geq \boldsymbol{\beta}$ for some $\boldsymbol{\lambda}$. Ben-Israel shows that we can also write the above linear program as

$$\begin{aligned} \max \quad & (F_{\mathbf{c}}^{-1}(\boldsymbol{\beta}))^T\mathbf{x} && (2.56)\\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq F_{\mathbf{b}}^{-1}(1-\boldsymbol{\alpha}) && (2.57)\\ & \mathbf{x} \geq 0, && (2.58) \end{aligned}$$

with $F_{\mathbf{c}}^{-1}(\boldsymbol{\beta})$ is defined similarly to $F_{\mathbf{b}}^{-1}(1-\boldsymbol{\alpha})$ above.

We capture uncertainty in the cost function using the expected values of the $\mathbf{c}$ vector, or by using quantiles of $\mathbf{c}$ that we want to protect against, and uncertainty in the RHS can be captured by using the relevant quantiles of the $\mathbf{b}$-vector.

Our focus in this work is on models whose objective function is to optimize the expected value of the objective function, with uncertainty in the right-hand side values, and hence we adopt the deterministic equivalent as shown in (2.53)-(2.55).

Thus, given the CDF of the right-hand-side (RHS) vectors, (or even certain *quantile* values of the distribution), we convert the stochastic problem into a deterministic linear programming problem of the same size as measured by the number of variables and constraints. Quantiles of the probability distribution for uncertain parameters can be obtained by analyzing historical data and incorporating additional knowledge of the system behavior.

The CCP model assumes the RHS ($= \mathbf{b}$) and $\mathbf{c}$ alone to be uncertain, and adjusts the values of these uncertain parameters to create a solution with more slack. The solution to the LP (2.53)-(2.55) has higher slack than the solution to (2.46)-(2.48). Unfortunately, chance-constrained programming encounters serious computational issues as we try to capture uncertainty in multiple coefficients per constraint. For most of their models, Charnes and Cooper limited uncertainty to one random variable per constraint (the $\mathbf{b}$-matrix value or RHS value). To incorporate uncertainty in the $\mathbf{A}$-matrix (left-hand-side), we must calculate a joint probability distribution for all uncertain coefficients in the constraint, making the deterministic program cumbersome to solve. Miller and Wagner [24] discuss chance-constrained methods for the case of multiple random variables (per constraint) generated by a multinomial distribution. However, most chance-constrained programming has been limited to uncertainty in the constraints only in the right-hand-side due to the difficulties associated with multiple random variables.

Modifying the right-hand side **b** vector in (2.52) is sufficient, however, to provide the entire constraint a protection of α_i. Therefore, though capturing uncertainty explicitly in the **A** matrix is cumbersome, constraint (2.52) is implicitly protecting, to some extent, against changes in the left-hand-side matrix.

The following are some of the **advantages** of this model:

- The structure of the CCP model is generalizable to all linear/integer programs.
- The model of capturing uncertainty has intuitive appeal. The deterministic formulation is also easy to understand and interpret.
- The CCP model does not require complete knowledge of the distribution that the uncertain data follows. In fact, knowledge of the quantile value of the distribution, corresponding to the required protection level for the constraint, is sufficient. In general, knowledge of a few discrete quantiles of the uncertain data for each constraint allows the user to approximate the distribution without requiring too much data about the distribution. Such information is also usually available through statistical analysis of the historical data of the system.
- Finer knowledge of the behavior of the system, as compared to simply the bounds of variation, can be captured through this model. Distributions other than the uniform distribution can be easily incorporated without an increase in complexity.

However, this model also has some **limitations**, as mentioned below:

- Uncertainty in the left-hand side **A** matrix, including correlations among uncertain data, is difficult to model explicitly.
- Approximate probability distributions or some quantiles of the distribution of the RHS have to be known. If unknown, the extreme-value bounds, as used in the Bertsimas-Sim model, can be considered as the bounds of a uniform distribution.

2.7 Extended Chance Constrained Formulation (ECCP)

The CCP faces a similar issue as the Bertsimas-Sim model: the problem of specifying a probability of satisfaction for each constraint. This is potentially a limitation of the approach when applying to large problems. We propose a model that avoids the need to specify the protection level for each constraint explicitly. Instead, we include a constraint on the overall expected profit of the robust solution and change the objective to one of maximizing the minimum protection level provided any constraint.

For this, we require the knowledge of some quantiles and their associated values, of the probability distribution of the RHS for each constraint, and the expected values of the profit function. Let K_i represent the set of quantiles known in the ith constraint. We define a binary variable α_i for each constraint i representing the quantile that is chosen from among the K_i available quantiles. p_i^k is the protection level probability associated with quantile $k \in K$ for constraint $i \in I$. The objective function value, denoted γ, equals the minimum protection level achieved over all constraints $i \in I$. To capture the tradeoff of robustness with profits, we assume that the planner is willing to forego a (user-specified) profit of Δ to instead gain a robust plan.

We propose a new budget-based chance-constrained formulation for (2.49)-(2.51) that can be written as:

$$\max \quad \gamma \tag{2.59}$$

$$\text{s.t.} \quad P(\mathbf{Ax} \leq \mathbf{b}) \geq \boldsymbol{\alpha} \tag{2.60}$$

$$\gamma \leq \alpha_i \qquad \forall\, i \in I \tag{2.61}$$

$$E(\mathbf{c}^T\mathbf{x}) \geq \mathbf{c}^T\mathbf{y}^* - \Delta \tag{2.62}$$

$$\mathbf{x} \geq 0 \tag{2.63}$$

$$\boldsymbol{\alpha} \geq 0 \tag{2.64}$$

Here $\mathbf{c}^T\mathbf{y}^*$ is the expected profit of the nominal optimal solution $\mathbf{y}^*$. γ indicates the minimum level of protection over the set I of all constraints i, and the objective is to maximize the minimum protection level over all constraints in the model. The objective may also be cast as a weighted sum of the protection levels of the constraints $= \sum_{i \in I} w_i \gamma_i$, with w_i representing the non-negative weight assigned to any constraint $i \in I$.

The formulation (2.59)-(2.64) is not linear. To write it in a linear fashion, we define the following notation:

Data:

- I: set of all constraints i;
- J: set of all variables j;
- K_i: number of discretized protection levels for constraint i;
- w_i: weight assigned to the protection level of the ith constraint;
- a_{ij}: constraint matrix coefficients of the problem, corresponding to the ith constraint and jth variable, $\forall\, i \in I, j \in J$;
- b_i: nominal RHS values corresponding to the ith constraint, $\forall\, i \in I$;
- c_j: expected profit coefficient corresponding to the jth variable, $\forall\, j \in J$;
- b_i^k: kth quantile value of the RHS parameter of the ith constraint, $\forall k \in K_i, i \in I$; and
- z_j^*: Optimal solution to (2.46) - (2.48) found using nominal values of the $\boldsymbol{b}$ and $\boldsymbol{c}$ parameters, $\forall\, j \in J$.

Decision variables:

- x_j: Non-negative decision variables in the problem, $\forall j \in J$

- y_i^k: Binary variable that is equal to 1 if the protection level (expressed as a probability p_i^k, with $0 \leq p_i^k \leq 1$) represented by the kth quantile ($k \in K_i$) is attained in constraint $i \in I$; and 0 otherwise. This means that if the kth quantile value is protected against, the $(k+1)$st quantile is also automatically protected against. This follows from the fact that constraints are "less than" inequalities. For example, if there are 5 quantile values for b, with $b \leq 2$ satisfied 1% of the time, $b \leq 3$ satisfied 5% of the time, $b \leq 5$ satisfied 20% of the time, $b \leq 7$ satisfied 35% of the time and $b \leq 8$ satisfied 50% of the time, then the quantiles k are $k_1 = 2$ with protection level 99%, $k_2 = 3$ with protection level 95%, $k_3 = 5$ with protection level 80%, $k_4 = 7$ with protection level 65%, and $k_5 = 8$ with protection level 50%. That is, a protection level of 95% is achieved with $b = 3$ (and hence 80%, 65% and 50% protection levels are also achieved with $b = 3$). The y_i^k values for any constraint i, therefore, follow a step function as shown in Fig. 2-1.

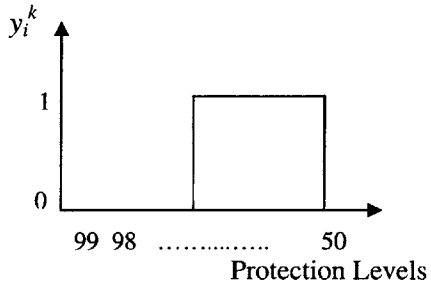

Figure 2-1: Step function representing the form of protection level variables

- γ_i: protection level attained for the ith constraint, $\forall\, i \in I$.

The extended chance-constrained model (ECCP) is as follows:

$$\max \quad \sum_{i \in I} w_i \gamma_i \tag{2.65}$$

$$\text{s.t.} \quad \sum_{j \in J} c_j x_j \geq \sum_{j \in J} c_j z_j^* - \Delta \tag{2.66}$$

$$\sum_{j \in J} A_{ij} x_j \leq \sum_{k=1}^{K} b_i^k (y_i^k - y_i^{k-1}) \quad \forall\, i \in I \tag{2.67}$$

$$y_i^k \geq y_i^{k-1} \quad \forall\, k = 1, ..., K_i, i \in I, \tag{2.68}$$

$$y_i^0 = 0 \quad \forall\, i \in I \tag{2.69}$$

$$y_i^K = 1 \quad \forall\, i \in I \tag{2.70}$$

$$\gamma_i \leq \sum_{k=1}^{K_i} p_i^k (y_i^k - y_i^{k-1}) \quad \forall\, i \in I \tag{2.71}$$

$$x_j \geq 0 \quad \forall\, j \in J \tag{2.72}$$

$$y_i^k \in \{0, 1\} \quad \forall\, k \in K_i, i \in I \tag{2.73}$$

$$0 \leq \gamma_i \leq 1 \quad \forall\, i \in I \tag{2.74}$$

The objective function (2.65) maximizes the weighted discretized probability that each constraint $i \in I$ is feasible. It can also be re-written to maximize the minimum value of γ_i over all constraints. (2.66) ensures that the solution's expected profit is within Δ units of the expected profit associated with the nominal optimal solution (found by solving the problem using nominal values of the **b** vector). For all constraints $i \in I$, (2.67) forces the left-hand-side (LHS) to be less than or equal to b_i^k if y_i^k equals 1, thereby ensuring constraint satisfaction with at least the probability associated with quantile k. For the smallest quantile k^* that can be satisfied, the $y_i^{k^*}$ value is 1, and quantiles $k < k^*$ have $y_i^k = 0$. Thus, the RHS value of this constraint is selected as the smallest one that can be satisfied by the solution. (2.68) ensures that the y_i^ks are monotonically increasing and follow the step function shown in Fig. 2-1, such that if a smaller quantile (higher protection) is achieved, the larger quantile (lower protection) is automatically achieved. (2.69) and (2.70) set the boundary values of the y_i^k step functions. Constraints (2.71) set γ_i to be no greater than the

highest protection level provided to constraint i by the solution. The x_js are non-negative for all $j \in J$; y_i^ks are binary for all $j \in J$ and all $k \in K_i$ for all $i \in I$; and γ_is are non-negative for all $i \in I$ as required by (2.72),(2.73) and (2.74), respectively.

2.7.1 Large-scale extension capabilities

In the ECCP, the number of constraints and variables is much greater than that of the original chance-constrained formulation. The additional constraints are needed to model the quantiles of each uncertain RHS parameter. Due to the nature of the step function, several of these constraints do not play a significant role, or are not *tight*, at the optimal solution. Thus, we can apply the concept of row-generation to make the solution process more tractable. Moreover, we can add quantiles dynamically, adding those corresponding to higher levels of protection only if the solution currently satisfies the highest level for a particular constraint. In doing so, we can greatly decrease the number of constraints in the model. By including only partial quantile information at first, we have a model with fewer constraints and variables that is easier to solve.

Because the number of decision variables also increases in the ECCP, the applicability of column generation techniques is critical. Hence, as in the case of the Delta model, we shall again write the standard multi-commodity flow problem ((4.1)-(4.4)) as an ECCP formulation and check if the property of column generation is retained.

Additional notation:

- L_{ij}: Set of quantiles $l = 1, ...|L_{ij}|$ of uncertain capacity parameters, for each constraint corresponding to arc (i, j).

- $f_p^{'*}$: Is the optimal solution to (4.1)-(4.4) using nominal values of data.

- p_{ij}^l: Is the protection level probability associated with quantile $l \in L_{ij}$ for the capacity constraint corresponding to arc (i, j), $0 \leq p_{ij}^l \leq 1$.

- y_{ij}^l: Is the binary variable that is equal to 1 if the protection level expressed as a probability p_{ij}^l, represented by the lth quantile, is attained in the capacity constraint for arc (i,j); and 0 otherwise.

- The duals of the constraints containing variable f_p are expressed in the rightmost column of each equation, with π^1 and π_{ij}^2 for all $(i,j) \in A$, non-negative and σ_k for all $k \in K$ unrestricted in sign.

The following formulation is analogous to (2.65) - (2.74).

$$\max \sum_{(i,j)\in A} w_{ij}\gamma_{ij} \tag{2.75}$$

$$\text{s.t.} \sum_{k\in K}\sum_{p\in P^k} d_k c_p f_p \geq \sum_{k\in K}\sum_{p\in P^k} d_k c_p f_p^{'*} - \Delta \qquad -\pi^1 \tag{2.76}$$

$$\sum_{k\in K}\sum_{p\in P^k} d_k f_p \delta_{ij}^p \leq \sum_{l=1}^{|L_{ij}|} u_{ij}^l (y_{ij}^l - y_{ij}^{l-1}) \qquad \forall\,(i,j) \in A \qquad \pi_{ij}^2 \tag{2.77}$$

$$y_{ij}^l \geq y_{ij}^{l-1} \qquad \forall\, l = 1, ..., |L_{ij}| \tag{2.78}$$

$$y_{ij}^0 = 0 \qquad \forall\,(i,j) \in A \tag{2.79}$$

$$y_{ij}^{|L_{ij}|} = 1 \qquad \forall\,(i,j) \in A \tag{2.80}$$

$$\sum_{p\in P^k} f_p = 1 \qquad \forall\, k \in K \qquad \sigma_k \tag{2.81}$$

$$\gamma_{ij} \leq \sum_{l=1}^{|L_{ij}|} p_{ij}^l (y_{ij}^l - y_{ij}^{l-1}) \qquad \forall\,(i,j) \in A \tag{2.82}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K \tag{2.83}$$

$$y_{ij}^l \in \{0,1\} \qquad \forall\, l \in L_{ij}, \forall\,(i,j) \in A \tag{2.84}$$

$$0 \leq \gamma_{ij} \leq 1 \qquad \forall\,(i,j) \in A \tag{2.85}$$

(2.75) is the ECCP objective function that maximizes a weighted sum of protection levels over constraints with uncertain parameters. Constraints (2.76) limit the expected cost of the robust solution (due to non-service of shipments) to no less than the user-specified value of Δ less than the expected optimal cost when using nominal

parameter values. Constraints (2.77) find the highest protection level attainable for the uncertain parameters. Inequalities (2.82) set γ_{ij} to be no greater than the highest protection level provided to the capacity constraint corresponding to (i,j). (2.78) ensure that the protection level variables follow a step function, that is, if a higher level of protection is achieved, all lower levels of protection are also achieved. (2.79) and (2.80) set the boundary values of the step functions. Constraints (2.81) assign one path to each shipment. Constraints (2.83), (2.84) and (2.85) require the y_{ij}^l and f_p variables to be binary, and the probability protection level variables to lie between or at 0 and 1.

The reduced cost of the variable f_p from the above formulation is

$$0 - [-\pi^1 d_k c_p + \sum_{(i,j)\in A} \pi_{ij}^2 d_k \delta_{ij}^p + \sigma_k] \tag{2.86}$$

$$= d_k(\sum_{(i,j)\in A} \pi^1 c_{ij} \delta_{ij}^p - \sum_{(i,j)\in A} \pi_{ij}^2 \delta_{ij}^p) - \sigma_k \tag{2.87}$$

$$= d_k \sum_{(i,j)\in A} (\pi^1 c_{ij} - \pi_{ij}^2)\delta_{ij}^p - \sigma_k. \tag{2.88}$$

This is similar to the form of the reduced cost expression in the case of standard multi-commodity flows. Hence, we can rewrite the objective function as $\min - \sum_{(i,j)\in A} w_{ij}\gamma_{ij}$ and solve a shortest path problem on the network with arc costs $-\pi^1 c_{ij} + \pi_{ij}^2$. If the shortest path for commodity k has length less then σ_k/d_k, then the path has negative reduced cost and it is added to the formulation and the augmented LP is re-solved. If no such path exists, the current solution is optimal for the linear programming relaxation.

Column generation approaches are therefore, compatible with chance-constrained programming models, and we can combine the approaches of column generation and row-generation for large-scale linear programs. These can be integrated to form branch-and-price-and-cut algorithms for large-scale binary programs.

2.8 Comparative Analysis and Insights into Berstimas-Sim and CCP Models and Extensions

In this section, we will first compare the Bertsimas-Sim and ECCP models in terms of their approach to modeling uncertainty. Then we will examine how robust solutions from these approaches are evaluated, and their behavior with respect to different metrics.

Conventional robust optimization models often have associated data requirements that are excessive, requiring data that is either unknown or is so extensive that it cannot be incorporated into a tractable model. Techniques such as the Bertsimas-Sim approach, simplify the data requirements, but do not take advantage of information, if known, pertaining to the distributions of uncertain data. The CCP/ECCP approaches can capture data distributions if known, and can work with extreme values such as those used by the Bertsimas-Sim model if distributions are unknown.

The assumption of a uniform symmetric distribution, as in the Bertsimas-Sim model, may not always be valid. If we provide protection against measurement error, it is more probable that the distribution of uncertainty is Gaussian, in which case the extreme values usually have a very low probability [28]. Protecting against such values is also overprotective and often expensive. Moreover, uncertainty distributions often tend not to be symmetric. For example, capacity shortages are much more common than capacity excesses.

The advantage of the CCP/ECCP approaches are that we can utilize knowledge of distributions and/or use several quantiles, and model with higher fidelity if desired. However, currently, we can incorporate uncertainty only in the objective function and RHS, but not in the $\mathbf{A}$ matrix without adding a lot of solution complexity. (To a certain extent we are protecting against some uncertainty in the A matrix by adjusting the $\mathbf{b}$ vector.) The Bertsimas-Sim approach, however, is general enough to model uncertainty in the $\mathbf{A}$ matrix also.

The protection level metrics for CCP/ECCP are more intuitively meaningful with respect to robustness than the measure of robustness used in the Bertsimas-Sim approach. Because the protection parameter for each constraint γ_i is apriori less meaningful, the Bertsimas-Sim models might have to be solved multiple times to identify the desired robustness levels. In contrast, the CCP models are intuitively more understandable. The extended versions of these models alleviate the need to provide protection level targets for each constraint and instead, achieve the 'highest' levels of protection for a given budget. While providing enhanced modeling capabilities, the performance of the solutions are best evaluated *not* by the objective function values but rather by other means such as simulation.

For the Bertsimas-Sim approach, row-generation and column generation are difficult or impossible to do efficiently. In contrast, row and column generation can be efficiently accomplished in the Delta, CCP and ECCP approaches. If the nominal problem formulation allows for column generation, the robust formulations of the Delta, CCP and ECCP models preserve this property, whereas the robust formulations of the Bertsimas-Sim model does not. This provides Delta, CCP and ECCP a definite advantage, one that is often necessary, in solving large-scale problems.

Several metrics can be used to evaluate the solutions obtained from robust optimization approaches. Unlike deterministic approaches in which the metrics are typically restricted to a comparison of the objective function values and the run times, there are several robustness metrics. These include quantification of the degree of violation of constraints; of how often constraints might be violated; of the scope of replanning required (the replanning may be local or global); on the number of times replanning is necessary; and so on. In 2006, Sakamoto [28] applied the Bertsimas-Sim model to a UAV mission planning problem. In this work, he developed a UAV Mission Planner that couples the scheduling of tasks with the assignment of these tasks to UAVs, while maintaining the characteristics of longevity and efficiency in the plans. The problem is formulated using the Bertsimas-Sim approach and is a mixed integer

program (MIP) that is evaluated using simulation. The author uses several metrics such as the number of constraints violated, the degree of constraint violation, and the time until violation of the first constraint to evaluate the true robustness of the solution. In this work, the author concludes that solutions obtained using nominal values for all parameters possess a degree of slack, especially due to the requirement of integer solutions. He concludes, however, that robustness modeling indeed adds value to the overall plan. He observes through simulation, however, that the number of violations did not decrease monotonically with increases in the protection levels. In a seemingly contradictory result, he concluded that increasing protection in the Bertsimas-Sim model did not increase the expected plan value to the user. Also, among the several metrics used to evaluate robustness, it was observed that increased protection rarely resulted in the simultaneous improvement of all metrics, indicating a tradeoff even among different metrics of robustness.

The Bertsimas-Sim model also did not produce meaningful objective function values, because the objective function costs themselves were 'protected against'. The result was that the objective function values ceased being good estimates of plan reward, suggesting the need for alternative means of protection for the objective function itself. The Bertsimas-Sim approach is most appropriate for uncertainty distributions that have non-negligible weight at the tails, for example, the uniform distribution. However in the case of Gaussian-type distributions, the applicability of the model must be investigated. Moreover, uncertainty correlations between constraint types and correlations between data uncertainties are not captured in this model.

Bryant [11] finds robust plans for the Effects-Based Operations (EBO) model of UAV mission planning. He applies both the models of Bertsimas-Sim and CCP to the problem. He finds that the robust plans in both cases are of far greater value than the deterministic plan with nominal values. He states that both the models have near-identical effects in terms of value added due to protection/robustness (in spite of the CCP model protecting only in the RHS, and the Bertsimas-Sim model in both

the RHS and LHS). As in the case of Sakamoto, he reports that the frequency of constraint violations does not decrease monotonically with protection level. He also observed through simulation that the CCP approach tends to produce better plans than the Bertsimas-Sim model in the case of normally distributed uncertainty.

Both authors agree that robustness gained by adding slack is seen to add value to solutions. Another general conclusion in the literature is that the ability to select gradations of the level of protection is of value in that it adds flexibility to the modeling process and helps tailor the solution to the needs of the user. However, it is not necessarily correlated with solution quality.

From the literature seen in this chapter, we see that there is a need to *define* suitable metrics for robustness. Because robustness metrics can be conflicting, it is important to decide which metrics are of primary importance.

Overall, we conclude that ECCP is the approach that meets our primary requirements of flexibility in using available information, ease of interpretation of robustness metrics, and scalability of the solution approach to large problems.

2.9 Summary

Robust optimization is a new field of study with great potential for impact. Literature in this field is not vast, especially in the case of large-scale problems. We examined different approaches to robust optimization in this chapter and found that the Bertsimas-Sim and the CCP approaches were the most applicable models given their tractability. Further in-depth analysis indicated the need for modeling enhancements, which we tailored for large-scale problems and detailed in sections 2.5 and 2.7. We also analyzed the scalability of these models, their flexibility (in terms of data requirements) and the interpretability of the inputs and the results. In spite of some disadvantages, the ECCP model was found to be more versatile than the Bertsimas-Sim model in applications to large-scale resource allocation problems.

Chapter 3

Modeling Shipment Routing With Time-Windows Under Uncertainty

In this chapter, we will address the problem stated in Chapter 1, and discuss different ways of modeling the shipment routing problem with time-windows under uncertainty (SRTW-UU), an instance of MCFTW-UU. We will apply some of the lessons from our examination of the Bertsimas-Sim and CCP models in Chapter 2 to the SRTW without consideration of elements of uncertainty and then expand our consideration to the SRTW-UU. We present several models and discuss their shortcomings. We then motivate the necessity for a new approach to solve both nominal problems as well as problems under uncertainty, and describe the framework of such an approach.

In the SRTW-UU, we are interested in addressing the stochastic nature of input data as seen in operations costs, vehicle capacities, shipment demands and supply quantities, and travel and service times. We formulate the problem using existing modeling approaches, and model uncertainty using the methods of Bertsimas-Sim and CCP. We shall show that the generalizable approaches of Bertsimas-Sim and CCP face issues in terms of scalability and capturing uncertainty. Problems involving commodity routing and vehicle scheduling are NP-hard because they generalize the Traveling Salesman Problem (TSP), which is known to be NP-hard [29]. Very large

instances of this problem have been addressed using heuristics because exact methods are not tractable [18]. To address problem size and tractability issues, we describe a new decomposition approach for the SRTW and SRTW-UU, and provide a detailed description of its design in Chapter 4.

3.1 Shipment Routing with Time-windows under Uncertainty (SRTW-UU): Modeling Approaches

Models for the SRTW have primarily been derived from the basic multi-commodity flow formulation. Modeling and solving multi-commodity flows has been addressed extensively in the literature, and standard methods exist to solve it [2]. For SRTW, additional decision making in the form of the vehicle scheduling element, along with the shipment routing element leads to a considerable increase in complexity. The scheduling element can be captured in terms of a *static* network representation or a *dynamic* network representation, as discussed in the following sections. We explore both static and dynamic modeling approaches in the context of uncertainty, by formulating them using the approaches of Bertsimas and Sim and CCP. We then discuss the relative advantages and disadvantages.

3.1.1 Static Network Approach - Schedule Modeled as a Continuous Variable

Network Description

The network $G = (N, A)$ with node set N and arc set A, for the continuous variable approach, as shown in Figure 3-1, is defined such that each node j of the network has three attributes: a location $l(j)$, vehicle $v(j)$ ($v(j) \in V$, the set of vehicles), and information if it is an arrival node or a departure node. An arrival node j corresponds to the arrival of vehicle $v(j)$ at location $l(j)$. Similarly a departure node j corresponds to the departure of vehicle $v(j)$ from location $l(j)$. For example, in Figure 3-1, node D_{A1} is the departure node for vehicle 1 at location A, and A_{A2} is the arrival node

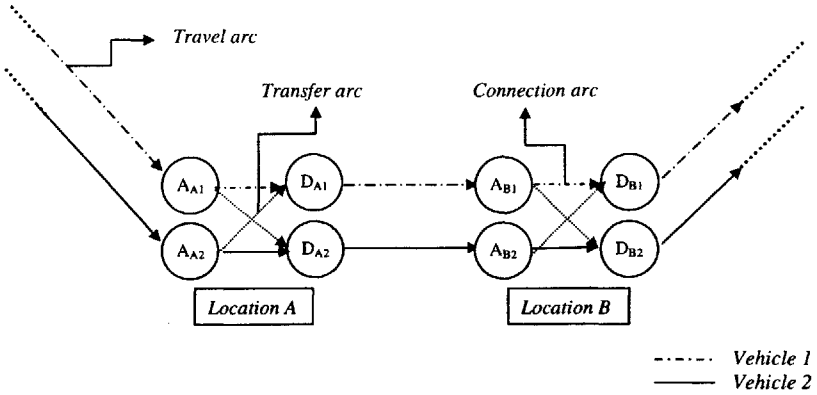

Figure 3-1: Network for the schedule-as-a-continuous-variable approach

for vehicle 2 at location A. The set of arcs A is partitioned into three arc sets, namely travel arcs, connection arcs and transfer arcs. *Travel arcs* (i, j) on the network represent movement of vehicle $v(i)(= v(j))$ departing from location $l(i)$ and arriving at location $l(j)$. Flows on these arcs represent the movement of shipments on vehicle $v(i)$ from $l(i)$ to $l(j)$. *Connection arcs* connect the arrival node of vehicle $v(i)$ at location $l(i)$ and the departure node of $v(i) = v(j)$ at location $l(j) = l(i)$. Flows on these arcs represent shipments remaining on vehicle $v(i)$ while it is positioned at location $l(i)$. We denote the set of travel arcs and connection arcs for vehicle v as $A(v)$, and the nodes that arcs in $A(v)$ are incident on as $N(v)$, for all $v \in V$. *Transfer arcs* (i, j) connect the arrival node of vehicle $v(i)$ at location $l(i)$ to the departure node of vehicle $v(j)(\neq v(i))$ at location $l(j) = l(i)$. Flow on these transfer arcs, denoted $A(t)$, represents the transfer of shipments between vehicles at a location. Associated with each transfer arc $(i, j) \in A(t)$ and shipment $k \in K$ is a transfer time tr_{ij}^k. Travel arcs, connection arcs and transfer arcs are shown in Figure 3-1. In addition, we have an artificial arc for each shipment $k \in K$, connecting the departure node of the shipment's origin location, denoted $O(k)$, to the arrival node of its destination location, denoted $D(k)$. We denote the set containing the artificial arcs for shipment k as A'_k and the set of artificial arcs for all shipments $k \in K$ as A'. The artificial arcs have infinite capacity and zero travel time, but have large cost equal to that associated with non-service of shipments. Hence, an artificial arc is used when we

cannot find a feasible path for the shipment, in which case it does not receive service.

In Chapter 1, we introduced the problem and associated notation as follows: d_k units of each shipment k in the set of shipments K must flow from $O(k)$ to $D(k)$ in the network, satisfying restrictions imposed by its earliest available time $EAT^k_{O(k)}$ at the origin and latest delivery time $LDT^k_{D(k)}$ at the destination. To formulate this problem over the static network, we introduce the following additional notation:

Additional Notation:

- K: Set of shipments, $k = 1, 2, ..., |K|$. In addition, k can take a value of 0, where the 0th commodity represents vehicle flows over the network $G = (N, A)$, and commodities $k = 1, 2, ..., |K|$ represent shipments $k = 1, 2, ..., |K|$.
- b^k_i: origin-destination indicator for shipment $k \in K$ at node $i \in N$, equals 1 at $O(k)$, -1 at $D(k)$, and 0 otherwise.

Uncertain Data:

- tt^k_{ij} : Time required for shipment k to traverse arc $(i,j), \forall\ (i,j) \in A$. This is equal to the travel time between the locations for vehicle v for travel arcs $(i,j) \in A(v)$; equal to the minimum turn time of vehicle v for connection arcs $(i,j) \in A(v)$; and equal to tr^k_{ij}, the time to transfer shipment $k \in K$ from node i to node j, for transfer arcs $(i,j) \in A(t)$.
- c^k_{ij}: If (i,j) is an artificial arc that connects $O(k)$ and $D(k)$, c^k_{ij} equals the cost (penalty) of non-service of shipment k, and equals a very large value M for all other shipments $l \in K, l \neq k$. This effectively disallows a shipment l from flowing on an artificial arc belonging to shipment k, because that indicates an infeasible path. If (i,j) is not an artificial arc, $c^k_{ij} = 0\ \forall\ k \in K$
- u_{ij}: capacity of arc $(i,j) \in A$
- M: A large positive number

Variables:

- y_{ij}^k : binary variable that takes value 1 if shipment $k \in K$ travels on arc (i,j), 0 otherwise. For the commodity $k = 0$ representing vehicle flows, we set $y_{ij}^0 = 1$ for all $(i,j) \in A(v)$.
- t_i: continuous variable representing the time at which vehicle v departs from node $i \in N(v), \forall\, v \in V$

Nominal Formulation

In the nominal model, we assume that all input parameters are known and invariant. The mathematical formulation for the problem is as follows.

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k y_{ij}^k \tag{3.1}$$

$$\text{s.t.} \quad t_i + tt_{ij}^k y_{ij}^k \leq t_j \qquad \forall\,(i,j) \in A, \forall\, k = 0,1,...,|K| \tag{3.2}$$

$$t_{O(k)} \geq EAT_{O(k)}^k (1 - y_{O(k),D(k)}^k) \qquad \forall\, k = 0,1,...,|K| \tag{3.3}$$

$$t_{D(k)} \leq LDT_{D(k)}^k + M(y_{O(k),D(k)}^k) \qquad \forall\, k = 0,1,...,|K| \tag{3.4}$$

$$\sum_{j:(i,j) \in A} y_{ij}^k - \sum_{j:(j,i) \in A} y_{ji}^k = b_i^k \qquad \forall\, i \in N, \forall\, k = 0,1,...,|K| \tag{3.5}$$

$$\sum_{k \in K} d_k y_{ij}^k \leq u_{ij} \qquad \forall\,(i,j) \in A \tag{3.6}$$

$$y_{ij}^k \in \{0,1\} \qquad \forall\,(i,j) \in A, \forall\, k = 0,1,...,|K| \tag{3.7}$$

$$t_i \geq 0 \qquad \forall\, i \in N \tag{3.8}$$

The objective function (3.1) minimizes the costs of penalties incurred due to non-service of shipments. Constraints (3.2) guarantee that the schedule associated with vehicle travel is feasible (given that $k = 0$ represents vehicle flows), and that shipment flows and schedules are feasible with respect to travel times, connection times, and transfer times. (3.3) and (3.4) disallow shipments that are served (not those assigned to the artificial arcs) from being picked up or dropped off before or after their earliest

available and latest delivery times, respectively. Constraints (3.5) maintain shipment flow balance at nodes. (3.6) are the vehicle and transfer capacity constraints that ensure satisfaction of capacity restrictions. (3.7) restrict the y_{ij}^k variables to values of 0 and 1, thereby disallowing shipments to be served along more than one path. (3.8) maintain the non-negativity of the schedule times.

The above formulation contains the arc-based multi-commodity flow constraints (3.5), (3.6) and (3.7), along with additional constraints (3.2)-(3.4) and (3.8) that ensure schedule feasibility.

This formulation contains $|K||A|$ binary variables, $|N|$ continuous variables and $|A| + |K|(2 + |N| + |A|)$ constraints; a very large formulation that is difficult if not impossible to solve for many practical size problem instances. Thus, even when uncertainty is not considered, the problem is difficult to solve for large instances. Additionally, if we model uncertainty using the approaches of Bertsimas-Sim and CCP or their extensions, we exacerbate the tractability issues associated with solving this formulation.

Due to the difficulties associated with solving large problems with the arc-based formulation, path-based multi-commodity flow approaches have been extensively studied and applied. A path-based formulation for the SRTW, which models shipment flows on paths rather than arcs, is formulated using the following notation.

Notation:

- c_p: cost of path $p \in P^k$, is equal to the penalty for non-service on each artificial path, and 0 for all other paths
- δ_{ij}^p: Arc-path indicator that is equal to 1 if arc (i, j) is on path p, and 0 otherwise
- P^k: The set of paths $p = 0, 1, ..., |P^k|$ connecting $O(k)$ and $D(k)$ for shipment $k \in K$, let $p = 0$ denote the path comprised of the artificial arc from $O(k)$ to $D(k)$.

- f_p: equals 1 if all d_k units of commodity k flow on any path $p \in P_k$; and equals 0 otherwise.

$$\min \quad \sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p \tag{3.9}$$

$$\text{s.t.} \quad t_i + \sum_{p \in P^k} tt_{ij}^k \delta_{ij}^p f_p \leq t_j \qquad \forall\, (i,j) \in A, \forall\, k = 0, 1, ...|K| \tag{3.10}$$

$$t_{O(k)} \geq EAT_{O(k)}^k (1 - f_0^k) \qquad \forall\, k = 0, 1, ..., |K| \tag{3.11}$$

$$t_{D(k)} \leq LDT_{D(k)}^k + M(f_0^k) \qquad \forall\, k = 0, 1, ..., |K| \tag{3.12}$$

$$\sum_{p \in P^k} f_p = 1 \qquad \forall\, k = 0, 1, ..., |K| \tag{3.13}$$

$$\sum_{k \in K} \sum_{p \in P^k} d_k \delta_{ij}^p f_p \leq u_{ij} \qquad \forall\, (i,j) \in A \tag{3.14}$$

$$f_p \in \{0, 1\} \qquad \forall\, p \in P^k, \forall\, k = 0, 1, ..., |K| \tag{3.15}$$

$$t_i \geq 0 \qquad \forall\, i \in N \tag{3.16}$$

Constraints (3.13) - (3.15) form the path-based multi-commodity flow formulation that can be solved using column generation approaches designed specifically for large-scale problems. However, the additional schedule related constraints (3.10) - (3.12) exceed the number of constraints in the path-based multi-commodity flow formulation, making solution, especially for large-scale problems, difficult if not impossible. For large-scale multi-commodity flows involving time, approaches have been developed and used that incorporate the time element into the network structure, and eliminate the need to include constraints of the form (3.10) - (3.12). The time-expanded network is called a *time-space network*. We discuss path-based time-space network models in the next sections.

3.1.2 Dynamic Network Approach - Time-space Networks

In this section, we model the SRTW-UU using a time-discretized *dynamic* approach. We will first present the nominal model for the problem.

In time-space networks, both time and space are captured together, as two dimensions, by using the same variable, as shown in Figure 3-2. The nodes represent locations at points in (or windows of) time and the arcs represent movement between locations over time.

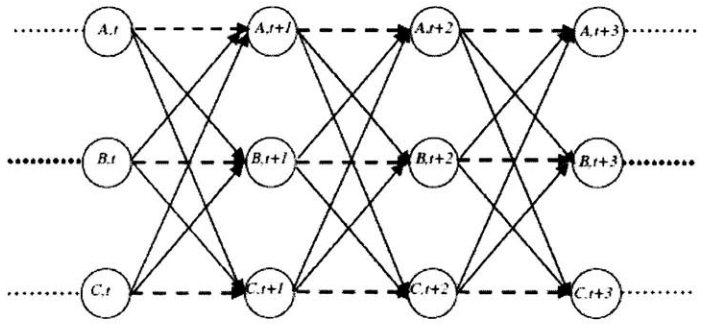

Figure 3-2: Example of a time-space network

This definition allows the SRTW to be formulated as a standard multi-commodity flow problem, without adding constraints for timing restrictions.

Network Description

The static network when expanded in time gives rise to the time-space network. Each node j of the time-space network $G = (N, A)$ is associated with a location $l(j)$, time $t(j)$, vehicle $v(j)$, and designation as an arrival node or a departure node. An arrival node j corresponds to the arrival of vehicle $v(j)$ at location $l(j)$ at time $t(j)$. Similarly a departure node j corresponds to the departure of vehicle $v(j)$ from location $l(j)$ at time $t(j)$. Arcs on the network represent movements between nodes. *Travel arcs* on the network represent movement of a shipment $k = 1, ...|K|$ or vehicle $v(i)(= v(j))$ departing from location $l(i)$ at time $t(i)$ and arriving at location $l(j)$ at time $t(j) \geq t(i) + tt_{ij}^{k}$. *Connection arcs* connect the arrival node of vehicle $v(i)$ at location $l(i)$ and time $t(i)$ and the departure node of vehicle $v(i) = v(j)$ at location

$l(i) = l(j)$ and time $t(j)$, allowing a shipment $k = 1, ...|K|$ to stay on a vehicle $v(i)$ at location $l(i)$. These arcs indicate that vehicle $v(i)$ remains at $l(i) = l(j)$, from time $t(i)$ to $t(j) \geq t(i) + tt_{ij}$. *Transfer arcs* $(i,j)_k$ connect the arrival node of vehicle $v(i)$ at location $l(i)$ and time $t(i)$ to the departure node of vehicle $v(j)(\neq v(i))$ at location $l(j)(= l(i))$ and time $t(j)$, such that $t(j) \geq t(i) + tt_{ij}^k$, allowing shipment $k = 1, ...|K|$ to transfer from $v(i)$ to $v(j)$ at location $l(i)$. In addition, we have artificial arcs $(l(O(k)), l(D(k)))$ from the origin node to the destination node of each shipment with infinite capacity and cost M, equal to a very large number.

Because the sequence of movements of each vehicle is known, but the schedule is unknown, we create departure and arrival nodes at each of the locations of each vehicle at all the possible (discrete) times that the vehicle can depart or arrive from the location. This allows us to create travel arcs that represent movements of the vehicle between the same locations, but at different times. Such travel arcs are *copies* of each other. The choice of a different copy indicates the choice of a different schedule. These copies are made such that the vehicle departs from its origin and reaches its destination within its specified time-window. In the solution, only one of the copies may be chosen in defining the schedule.

In this network, a vehicle route is simply an alternating sequence of travel arcs and connection arcs, with the timing of each connection arc satisfying minimum time requirements for a vehicle to stopover at a location. For each vehicle, several paths exist in the network, each representing the single route of the vehicle but with different associated schedules. Among all the schedules for each vehicle, one and only one may be chosen and the selected schedule must minimize the costs incurred for shipment non-service.

A feasible shipment path is an alternating sequence of travel arcs and connection or transfer arcs, with all shipment time-windows satisfied.

We formulate this problem by defining the variables corresponding to vehicle routes that obey the vehicle time-windows and shipment routes that obey shipment time-windows. We introduce the following additional notation and provide our formulations as follows:

Data:

- V: set of vehicles
- R_v: set of feasible paths on the network for vehicle $v \in V$
- ζ_{ij}^r: is 1 if arc $(i,j) \in A$ is included in vehicle route $r \in R$; and 0 otherwise
- δ_{ij}^p: is 1 if arc $(i,j) \in A$ is included in shipment path p; and 0 otherwise
- u_v: capacity of vehicle v

Shipment paths $p \in P^k, \forall\ k \in K$ and vehicle paths $r \in R_v$ for vehicle v are picked such that they satisfy the scheduling constraints (3.2)-(3.4) and (3.8). Thus, by definition, the paths chosen have a feasible schedule with respect to the nominal values of travel, connection and transfer times. We call this network the *nominal time-space network*. Therefore, only the demand satisfaction constraints and capacity constraints have to be explicitly formulated.

Variables:

- f_p: is a binary decision variable whose value is 1 if shipment path p is present in the solution, and 0 otherwise
- y_r: is a binary decision variable whose value is 1 if vehicle route r is present in the solution, and 0 otherwise

Nominal Formulation

$$\min \quad \sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p \tag{3.17}$$

$$\text{s.t.} \quad \sum_{p \in P^k} f_p = 1 \qquad \forall\, k \in K \tag{3.18}$$

$$\sum_{r \in R_v} y_r = 1 \qquad \forall\, v \in V \tag{3.19}$$

$$\sum_{k \in K} \sum_{p \in P^k} d_k \delta_{ij}^p f_p \leq \sum_{v \in V} u_v \sum_{r \in R_v} \zeta_{ij}^r y_r \qquad \forall\, (i,j) \in A \tag{3.20}$$

$$y_r \in \{0,1\} \qquad \forall\, r \in R_v, \forall\, v \in V \tag{3.21}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K \tag{3.22}$$

The objective function (3.17) minimizes the expected penalty costs of the shipments that are not serviced. Constraints (3.18) choose exactly one path for each shipment amongst each of its alternative paths. Equalities (3.19) choose exactly one route among all the vehicle paths on the network for each vehicle. Because the sequence of stops the vehicle makes is known, this is equivalent to choosing exactly one schedule for each vehicle. Inequalities (3.20) constrain the flow on each arc (i,j) to its possible capacity if a vehicle were assigned. (3.21) and (3.22) restrict the vehicle schedule choice variables and the shipment path choice variables to be binary.

The formulation (3.17) - (3.22) selects network paths for both the vehicles and the shipments, and in doing so, determines vehicle schedules and shipment paths and schedules simultaneously. In this process, we make decisions regarding the presence of arcs in the solution. In that sense, we are solving a network design problem, which we described in Chapter 1.

The number of constraints is $|K| + |V| + |A|$ and the number of variables in the formulation is equal to the total number of possible schedules for vehicles and possible paths for shipments on the network. In general, compared to the formulation (3.1) -

(3.8), we will have far fewer constraints, but many more variables. In fact, for large problems, because the number of path variables is exponentially large, the number of variables is so large that they cannot be enumerated fully. However, as described in Section 2.2, iterative techniques such as row and column generation can be used to solve large-scale problems of this type. Applicability of such large-scale techniques is critical to being able to achieving optimal or near-optimal solutions to the nominal formulation.

Robust Formulations

In (3.17) - (3.22), constraints (3.20) have uncertain parameters. In order to capture uncertainty in the shipment demands and supplies and vehicle capacities, we can apply the Bertsimas-Sim and CCP models and their extensions. As discussed in Chapter 2, ECCP is the approach we will implement to solve SRTW and SRTW-UU because we can capture information regarding the distribution of supply, demand, and capacity uncertainty, can apply column and row generation techniques, and do not have to specify the target probability of satisfaction for each individual constraint.

Constraints (3.20) contain uncertain parameters in both the right-hand-side and the left-hand-side of the constraint, with the demand parameters on the left-hand-side and the capacity parameters on the right-hand-side. However, correlations between both these uncertain parameters are unlikely, which avoids excessive complications due to usage of multinomial distributions as seen in [24]. According to (2.52), we see that using a quantile value of the right-hand side protects the entire constraint against uncertainty, thus protecting partially against uncertainty in the left-hand-side parameters. We can therefore use ECCP to provide protection against uncertainty in constraints of the form (3.20). We formulate (3.17) - (3.22) as an ECCP using the following additional notation.

Additional notation:

- L_{ij}: Set of quantiles $l = 1, ...|L_{ij}|$ of uncertain capacity parameters, for each

constraint corresponding to arc (i,j)

- f_p^*: Is the optimal solution to (3.17)-(3.22) using nominal values of data.
- p_{ij}^l: Is the probability protection level ($0 \leq p_{ij}^l \leq 1$) associated with quantile $l \in L_{ij}$ for arc (i,j)
- z_{ij}^l: binary variable that is equal to 1 if the protection level represented by the lth quantile is attained in the capacity constraint of arc (i,j); and 0 otherwise.
- γ_{ij}: maximum protection level attained for the capacity constraint of arc (i,j), for all $(i,j) \in A$, with γ_{ij} expressed as a probability.
- w_{ij}: non-negative weight assigned to the protection level of the capacity constraint corresponding to arc (i,j).

Robust Formulation with the ECCP approach

$$\max \quad \sum_{(i,j)\in A} w_{ij}\gamma_{ij} \tag{3.23}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{p\in P^k} d_k c_p f_p \leq \sum_{k\in K}\sum_{p\in P^k} d_k c_p f_p^* + \Delta \tag{3.24}$$

$$\sum_{p\in P^k} f_p = 1 \qquad \forall\, k \in K \tag{3.25}$$

$$\sum_{r\in R_v} y_r = 1 \qquad \forall\, v \in V \tag{3.26}$$

$$\sum_{k\in K}\sum_{p\in P^k} d_k \delta_{ij}^p f_p \leq \sum_{l=1}^{|L_{ij}|} u_v^l \zeta_{ij}^v (z_{ij}^l - z_{ij}^{l-1}) \qquad \forall\,(i,j) \in A \tag{3.27}$$

$$z_{ij}^l \geq z_{ij}^{l-1} \qquad \forall\,(i,j) \in A, l \in L_{ij} \tag{3.28}$$

$$z_{ij}^0 = 0 \qquad \forall\,(i,j) \in A \tag{3.29}$$

$$z_{ij}^{|L_{ij}|} = 1 \qquad \forall\,(i,j) \in A \tag{3.30}$$

$$\gamma_{ij} \leq \sum_{l=1}^{L_{ij}} p_{ij}^l (z_{ij}^l - z_{ij}^{l-1}) \qquad \forall\,(i,j) \in A \tag{3.31}$$

$$\sum_{k\in K}\sum_{p\in P^k} d_k\delta_{ij}^p f_p \leq \sum_{v\in V} M \sum_{r\in R_v} \zeta_{ij}^r y_r \quad \forall\,(i,j)\in A \tag{3.32}$$

$$z_{ij}^l \in \{0,1\} \qquad \forall\,(i,j)\in A, l\in L_{ij} \tag{3.33}$$

$$y_r \in \{0,1\} \qquad \forall\, r\in R_v, \forall\, v\in V \tag{3.34}$$

$$f_p \in \{0,1\} \qquad \forall\, p\in P^k, \forall\, k\in K \tag{3.35}$$

$$0 \leq \gamma_{ij} \leq 1 \qquad \forall\,(i,j)\in A \tag{3.36}$$

Constraints (3.25), (3.26), (3.32), (3.34) and (3.35) are the constraints in the nominal formulation, while (3.23), (3.24), (3.27), (3.28), (3.29), (3.30), (3.31), (3.33) and (3.36) are as presented in the ECCP formulation in Section 2.7.1.

Through this formulation, we have captured uncertainty in the shipment demands and supplies, and vehicle capacities, in such a way that column generation is possible. Additionally, if we require capture of uncertainty in the time element - the travel times, connection times and transfer times - we can do so by changing the network to incorporate time uncertainty. To do so, we re-draw the time-space network in the spirit of the CCP, with modified arc traversal times set equal to apriori decided quantiles of the distribution of traversal time for each arc. We refer to this modified time-space network as the robust time-space network. By solving the ECCP formulation (3.23) - (3.36) on this robust time-space network, we can simultaneously capture uncertainty in the travel times, shipments supplies and demands, and vehicle capacities.

Another way of capturing uncertainty in the travel times is to make copies of each arc in the nominal time-space network that capture different quantiles of the arc traversal times. This can capture a larger range of possible vehicle schedules and shipment paths than the network that specifies the quantiles of times apriori. However, it presents a formidable challenge in terms of the number of possible paths on the network, which increases the size of the problem far beyond that in the nominal time-space network.

Thus, using the ECCP approach, we can directly capture variability in the uncertain parameters, and solve it using large-scale approaches like column generation. But the size of the problem is now huge, due to additional size associated with capturing uncertainty. Hence, for large instances, the ECCP formulation presents a tremendous tractability challenge, one which we address by applying a new decomposition modeling and algorithmic approach that we have developed.

3.1.3 Motivation for a New Approach

As seen in Section 3.1, robust optimization approaches such as the Bertsimas-Sim and CCP and their extensions, do not scale well to the STRW-UU. This results in a likely inability to solve large-scale network-based resource allocation problems even when solution times are allowed to be long, and is certainly even more difficult when real-time operational solutions are required.

The challenge, therefore, is to design a model structure that can incorporate data related to parameter uncertainty, but remain tractable. A design goal is that the model's structure should remain the same, whether the data is known, partially known, or unknown. In the following sections, we will present such a modeling approach, and discuss how it overcomes the limitations of existing models while taking advantage of the strengths of the CCP and ECCP approaches, namely, flexibility with respect to data requirements and amenability to column generation.

3.2 Model Decomposition

3.2.1 Decomposition Overview

Our decomposition model leverages the natural advantages of the static as well as dynamic models. It builds off the static model by not considering the scheduling component in the network construction. It also seeks to use the efficient multi-commodity flow path formulation presented in the context of time-space networks, while avoid-

ing the intractability issues faced with such models. This leads us to consider a *decomposed* model in which routing and scheduling are treated separately.

Decomposition itself is used in several solution techniques, such as Lagrangean relaxation [2], Dantzig-Wolfe decomposition [2], Resource-directive decomposition [2], etc.

Our decomposition modeling approach separates the shipment flows, and scheduling (of vehicles and shipments) in the SRTW-UU problem into two modules - the flow module and the scheduling module. Decisions pertaining to shipment flows are made in the flow module and decisions regarding the shipment and vehicle scheduling are made in the scheduling module. Because the costs in the objective function for the problem can be expressed as purely flow-based costs (costs for using artificial arcs) we formulate the flow module as an optimization problem which we call the *Flow Master Problem.* The Flow Master Problem finds a cost-minimizing set of path flows for the shipments. Given shipment paths and vehicle routes, the *Scheduling Sub-problem* attempts to find a feasible schedule for the vehicle and shipment movements. It is possible, however, because scheduling constraints are not considered in the Flow Master Problem, that some of them are violated and a feasible schedule does not exist. In that case, in solving the Scheduling Sub-Problem, we identify a set of constraints to add to the Flow Master Problem to eliminate the current infeasible solution. These constraints 'cut' off the infeasible solution but not any schedule-feasible solutions in the Flow Master Problem. The Flow Master Problem and the Scheduling Sub-Problem are solved repeatedly until a feasible, and hence optimal, solution is identified by the iterative approach.

In the following sections, we will provide further details about the formulations of the Flow Master Problem and Scheduling Sub-Problem, focusing first on the nominal variants and later showing how to extend them to include uncertainties.

3.2.2 Flow Master Problem

For each shipment, we build a network similar to that described in Section 3.1.1 and Figure 3-1. A *shipment path* consists of a sequence of alternating travel and connection/transfer arcs from the shipment origin to its destination. Each sequence of arcs on this network does not contain specific scheduling information, although some ordinal time relations are implied. A schedule is the set of times at which the tasks along the shipment/vehicle route are performed. A *feasible schedule* is defined as one which allows both vehicles and shipments to travel along their paths while obeying all time-window constraints and making all the required connections and transfers.

In each of the shipment networks, there are several paths (including artificial paths, which consist of the artificial arc alone) to which the shipment may be assigned. For each shipment, exactly one such path has to be picked, and there must be a solution associated with the set of selected paths that satisfies capacity and scheduling constraints. Referring to the nominal formulation in Section 3.1.1, we see that constraints 3.2, 3.3, 3.4 and 3.8 are the scheduling constraints and 3.5, 3.6, 3.7 are the flow constraints.

When the scheduling constraints are relaxed, the problem reduces to flowing shipments on the vehicles. We formulate this as a path-based multi-commodity flow problem (with each shipment representing a commodity) so that techniques such as column generation can be used to solve it. Thus, the Flow Master Problem is an integer program that assigns one path to each shipment. We first present the nominal version of the problem and then its robust version.

The nominal version of the Flow Master Problem, denoted **SRTW-MP**, is defined over the static network presented in Section 3.1.1 and is formulated as:

$$\min \quad \sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p \tag{3.37}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{p \in P^k} d_k \delta_{ij}^p f_p \leq u_{ij} \quad \forall\, (i,j) \in A \tag{3.38}$$

$$\sum_{p \in P^k} f_p = 1 \qquad \forall\, k \in K \tag{3.39}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K. \tag{3.40}$$

The structure of this formulation is equivalent to the standard path-based multi-commodity flow formulation. Due to the removal of the scheduling decisions, the SRTW-MP is easier to solve than either the static network or dynamic network models presented in Section 3.1.

Formulation (3.37) - (3.40) can be modified to incorporate uncertainty in supplies, demands and capacities using the Extended Chance-Constrained model (ECCP) presented in Section 2.7. We capture uncertainty explicitly in capacities, and through the increased slack in the capacity constraint, also protect against demand uncertainty. By obtaining the maximum amount of slack possible through the utilization of the capacity quantile, we are effectively using a higher quantile of demand to protect against demand uncertainty.

Using the same notation for the ECCP models as used in Section 2.7.1, the ECCP formulation of the SRTW-MP, denoted SRTW-UU-MP, is as follows:

SRTW-UU-MP:

$$\max \quad \sum_{(i,j) \in A} w_{ij} \gamma_{ij} \tag{3.41}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p \leq \Big(\sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p'\Big)^* + \Delta \tag{3.42}$$

$$\sum_{k \in K} \sum_{p \in P^k} d_k f_p \delta_{ij}^p \leq \sum_{l} u_{ij}^l (y_{ij}^l - y_{ij}^{l-1}) \qquad \forall\, (i,j) \in A, \forall\, l \in L_{ij} \tag{3.43}$$

$$y_{ij}^{l} \geq y_{ij}^{l-1} \qquad \forall\, l \in L_{ij} \tag{3.44}$$

$$y_{ij}^{0} = 0 \qquad \forall\, (i,j) \in A \tag{3.45}$$

$$y_{ij}^{L} = 1 \qquad \forall\, (i,j) \in A \tag{3.46}$$

$$\sum_{p \in P^k} f_p = 1 \qquad \forall\, k \in K \tag{3.47}$$

$$\gamma_{ij} \leq \sum_{l=1}^{|L_{ij}|} p_{ij}^{l}(y_{ij}^{l} - y_{ij}^{l-1}) \qquad \forall\, (i,j) \in A \tag{3.48}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K \tag{3.49}$$

$$y_{ij}^{l} \in \{0,1\} \qquad \forall\, l \in L_{ij}, \forall\, (i,j) \in A \tag{3.50}$$

$$0 \leq \gamma_{ij} \leq 1 \qquad \forall\, (i,j) \in A \tag{3.51}$$

The objective function and constraints (3.41) - (3.51) are as presented in Section 2.7.1.

In spite of additional variables to capture quantiles, we know that the formulation is amenable to column generation (as detailed in Section 2.7.1) of the path variables, whose number is the same as that in networks involving multi-commodity flows without time-windows.

3.2.3 Scheduling Sub-Problem (SRTW-SP)

Given a shipment flow solution F from the SRTW-MP or SRTW-UU-MP, the objective of the Scheduling Sub-Problem (SRTW-SP) is to determine if the shipment flows obtained from solving the Flow Master Problem satisfy scheduling constraints (3.52)- (3.55), that is, if the flows have a feasible schedule obeying time-windows and allowing connections and transfers. All other constraints are contained within the SRTW-UU-MP, and therefore, are satisfied by F. If the scheduling constraints are not satisfied, F is not a feasible solution to the SRTW-UU.

$$t_i + \sum_{k \in K} tt_{ij}^{k} y_{ij}^{k} \leq t_j \qquad \forall\, (i,j) \in A \tag{3.52}$$

$$t_{O(k)} \geq EAT_{O(k)}^{k}(1 - y_{O(k),D(k)}^{k}) \qquad \forall\, k \in \{0\} \cup K \tag{3.53}$$

$$t_{D(k)} \leq LDT^k_{D(k)} + M(y^k_{O(k),D(k)}) \quad \forall\, k \in \{0\} \cup K \tag{3.54}$$

$$t_i \geq 0 \quad \forall\, i \in N \tag{3.55}$$

Solving the SRTW-SP can be accomplished by solving a series of shortest path and network-labeling algorithms. From the SRTW-UU-MP, shipment paths F and vehicle routes are known. We can determine if all the scheduling constraints are satisfied by finding feasible time-windows for each shipment at every node in the network using efficient network search algorithms. If we find a feasible schedule, we are done, otherwise, the same algorithm identifies the sources of infeasibility. We provide details of these infeasibilities, the means of identification of such infeasibilities, and incorporation of uncertainties in Chapter 4.

Uncertainty in travel times, connection times and transfer times is captured using quantiles of the travel times, connection times and transfer times that are chosen apriori, and reflect the protection level that we want to provide for each arc in the solution. We refer to the chance-constrained version of the SRTW-SP as the SRTW-UU-SP, and its form is exactly the same as the SRTW-SP. Whether solving SRTW-SP or SRTW-UU-SP, then, the procedure for identifying feasible schedules or infeasibilities remains the same. The only difference between the two is the values of traversal time on arcs. Due to the ease of solving SRTW-UU-SP, we can perform sensitivity analysis for different possible choices of apriori quantiles.

3.2.4 Iterative Feedback Mechanism

As shown in Figure 3-3, we iterate between solving the SRTW-UU-MP and the SRTW-UU-SP, identifying infeasibilities in the SRTW-UU-SP solutions and adding them as new constraints into the SRTW-UU-MP in order to eliminate current infeasibilities. We will show in Chapter 4 that these constraints eliminate exactly the infeasible solution but do not eliminate any feasible solution. With each iteration, the constraints added to the SRTW-UU-MP increase its size minimally, but decrease the size of its feasible solution space. Thus, we converge to the optimal solution with

each iteration of the algorithm. The rate of convergence can be improved if we can add cuts that eliminate more of the infeasible solution space in each step. Such effective cuts can be identified by finding cliques in the SRTW-UU-SP. We discuss this in further detail in Chapter 4. We have solved the SRTW-UU when the shipment flows obtained from the SRTW-UU-MP have associated feasible schedules, and the iterative procedure terminates. In Chapter 4, we will describes the algorithmic procedure in greater detail.

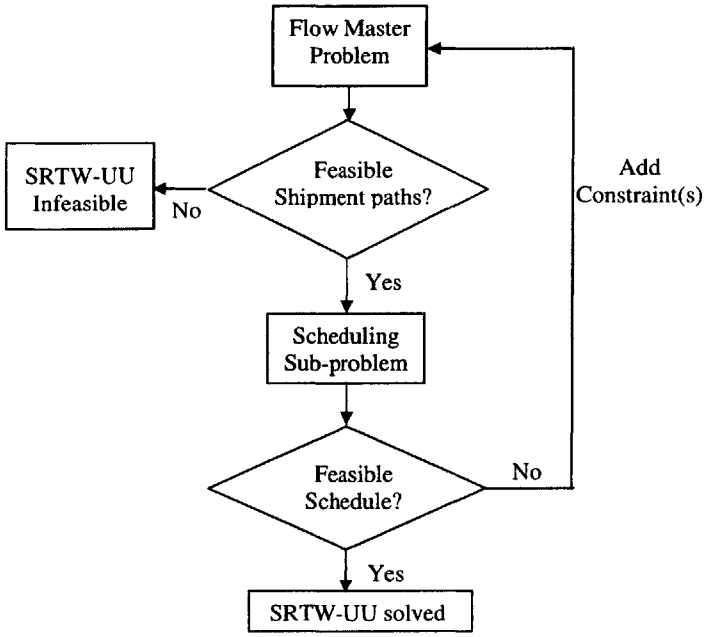

Figure 3-3: Iterative mechanism of the decomposition approach

3.3 Summary

In this chapter, we examined existing modeling methods for the SRTW-UU. Static network approaches as well as time-space network approaches encounter issues of tractability, especially for problem instances under uncertainty. For this reason, we propose a modeling approach where the problem is decomposed into flow (STRW-UU-MP) and scheduling (STRW-UU-SP) modules and solved separately. An iterative

approach repeatedly solves the two problems and terminates with an optimal solution to the unified flow and scheduling problem. We incorporate knowledge of uncertain data distributions and provide protection to the solution by modeling the STRW-UU-MP as an ECCP, thus capturing uncertainty in vehicle capacities, shipment supplies and demands, and limiting expected costs. We capture uncertainty in times using the scheduling sub-problem, and solve it using efficient network labeling algorithms. The combined structure, though iterative, ameliorates tractability issues due to the solvability of the multi-commodity flow problems without time-windows (the Flow Master Problem) relative to its counterpart with time-windows, and the ease of solving the Scheduling Sub-Problems.

Chapter 4

Solving Shipment Routing With Time-Windows Under Uncertainty

We described the model of our decomposition approach in Chapter 3. In this chapter, we illustrate the workings of our approach by means of an example. This example shows how decomposition works in the nominal case, that is, when data uncertainty is not considered. We then provide an algorithmic structure for solving the SRTW under more general cases involving uncertainty. Finally, we analyze the various elements of the decomposition algorithm, and provide insights for applying the approach to large-scale problems.

4.1 Illustration

We illustrate the workings of our decomposition approach in the nominal case using an example. Suppose the network shown in Figure 4-1 is a simplified version of the static network described in Section 3.2.2. Consider the network $G = (N, A)$ with costs and capacities on each of the arcs, and shipment origins, destinations and time-windows as described in Table 4.1. All arcs on this network have capacities of 2 units and unit travel times, and all connection times are 0.1 units. Each vehicle has a total of 8 hours to complete its shift, starting at time 0 and ending at 8. We build one such network for each shipment, and track the flow of the shipment on the network

through labels.

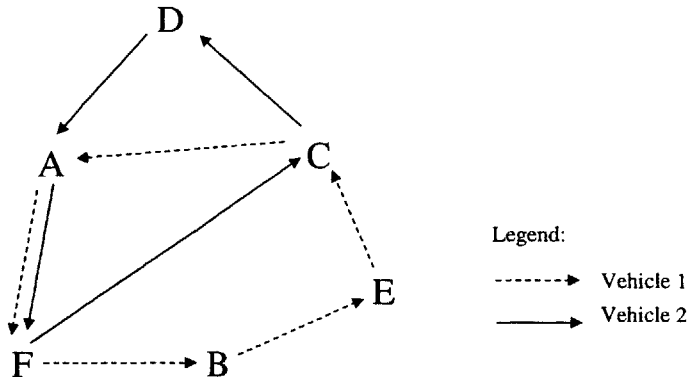

Figure 4-1: Network Characteristics

Shipment	Origin	Destination	Demand	EAT	LDT	Penalty
1	A	C	1	2	4.2	100
2	E	D	1	3.5	6	100
3	E	A	1	4.2	9	100

Table 4.1: Shipment Characteristics

Flow Master Problem

Each shipment has several network paths over which it can travel from its origin node to its destination node on the network. For example, shipment 1 has the following paths from its origin to its destination: $(AF)_1 - (FC)_2$, $(AF)_2 - (FC)_2$, $(AF)_1 - (FB)_1 - (BE)_1 - (EC)_1$, $(AF)_2 - (FB)_1 - (BE)_1 - (EC)_1$, where $(AF)_1 - (FC)_2$ indicates the path that travels from A to F on vehicle 1 and from F to C on vehicle 2. In the Flow Master problem, a path is assigned to each shipment by solving the following path-based multi-commodity flow formulation.

$$\max \quad \sum_{k \in K} \sum_{p \in P^k} d_k c_p f_p \tag{4.1}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{p \in P^k} d_k \delta_{ij}^p f_p \leq u_{ij} \quad \forall\, (i,j) \in A \tag{4.2}$$

$$\sum_{p \in P^k} f_p = 1 \qquad \forall\, k \in K \tag{4.3}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K. \tag{4.4}$$

After solving the above formulation, suppose the assigned paths are as follows:

Shipment 1: $(AF)_1 - (FC)_2$

Shipment 2: $(EC)_1 - (CD)_2$

Shipment 3: $(EC)_1 - (CD)_2 - (DA)_2$.

Note that shipments 2 and 3 travel together on leg EC of vehicle 1 and on leg CD of vehicle 2, requiring vehicles 1 and 2 to be at node C at the same time.

Scheduling Sub-Problem

In the Scheduling Sub-Problem module, either a feasible schedule for the shipment flows specified in solving the Flow Master Problem is determined, or the current Flow Master Problem solution is found to be schedule-infeasible. A feasible schedule is one that allows all the shipments and vehicles to travel on their respective paths while allowing sufficient time for connections and transfers to take place, while schedule-infeasible solutions do not allow sufficient time. Schedule infeasibilities are caused by paths that do not allow vehicles or shipments that have to travel together to be present in the same place at the same time. We determine if a feasible schedule exists by finding the earliest arrival time (EAT) and latest departure time (LDT) of each shipment at each node in the network. If the time-windows, determined as (EAT, LDT), are such that they allow shipments that have to connect or transfer across vehicles to be present in the same place at the same time, then a feasible schedule exists.

The first step in solving the Scheduling Sub-Problem is to find time-windows of vehicles and shipments at all nodes, by finding the shortest path for each shipment from its origin to all nodes in the network, and from all nodes to its destination. (The latter can be accomplished by reversing arc directions and running simple shortest path algorithms from the destination node to all other nodes.) For our example, we find the shipment time-windows in Table 4.2 and vehicle time-windows in Figure 4-3. If a vehicle or shipment cannot reach a node, its time-windows at the node will have negative duration, that is, the difference $LDT - EAT$ will be negative. Observe from Table 4.2 that shipment 1 cannot use a path passing through nodes D or E, shipment 2 cannot use a path passing through nodes A, B or F, etc. Because there are no possible paths through these nodes, we remove nodes D and E (and all incident arcs) from shipment 1's network and nodes A, B and F from shipment 2's network. Thus, after this operation, shipment 1 travels on the network shown in Figure 4-2. This process allows us to capture routing and scheduling infeasibilities.

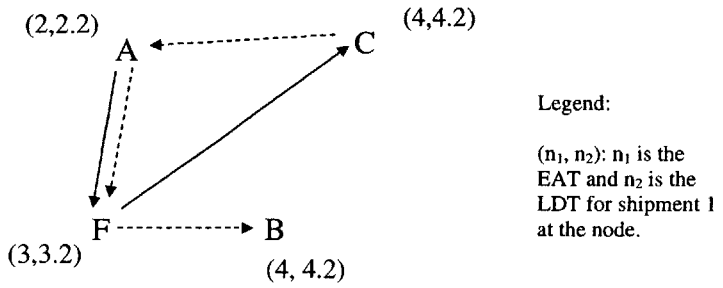

Figure 4-2: Network of Shipment 1

Importantly, note that routing and scheduling infeasibilities such as those described above, can be identified without knowledge of the paths assigned in the Flow Master Problem. Therefore, we eliminate such infeasibilities before solving the Flow Master Problem using a Pre-processing step that excludes paths that are schedule-infeasible.

Given the assigned paths for shipments 1, 2, and 3, in the next step of solving the Scheduling Sub-Problem, we determine if a feasible schedule exists, and if

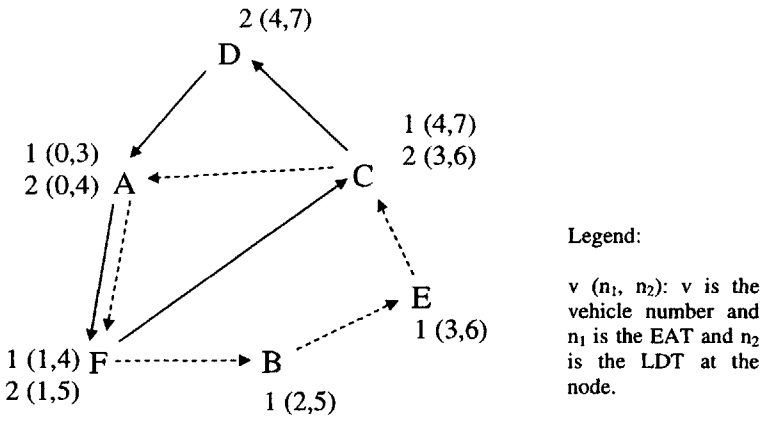

Figure 4-3: Vehicle Time Windows

	Node A	Node B	Node C	Node D	Node E	Node F
Shipment 1	(2, 2.2)	(4, 4.2)	(4, 4.2)	(5, 1.2)	(5, 3.2)	(3, 3.2)
Shipment 2	(5.5, 3)	(7.5, 3)	(4.5, 5)	(5.5, 6)	(3.5, 4)	(6.5, 2)
Shipment 3	(6.2, 9)	(8.2, 6)	(5.2, 8)	(6.2, 8)	(4.2, 7)	(7.2, 7)

Table 4.2: Shipment Time-Windows (EAT, LDT)

not, we identify the sources of the infeasibilities. We begin by re-computing the time-windows for each shipment, given the specific path to which it is assigned in the solution to the Flow Master Problem. We also compute the time-windows of each vehicle-shipment pair at all nodes of the network. For example, to compute a vehicle-shipment time-window for vehicle v and shipment s at node i, we let $EAT^i = \max\{EAT^i_v, EAT^i_s\}$, with EAT^i_v representing the earliest arrival time at node i for vehicle v and EAT^i_s representing the earliest arrival time for shipment s at node v; and $LDT^i = \min\{LDT^i_v, LDT^i_s\}$, with LDT^i_v representing the earliest arrival time at node i for vehicle v and LDT^i_s representing the earliest arrival time for shipment s at node i.

For each vehicle route or shipment path, there is an ordinal relationship between its nodes, with node i preceding another node j along the route or path. We refer to i as 'upstream' of node j and node j as 'downstream' of node i. Note that upstream

(vehicle-shipment)	(EAT,LDT)
(1,1)	(4, 4.2)
(1,2)	(4.5, 5)
(1,3)	(5.2, 7)

Table 4.3: Vehicle-Shipment Time Windows at Node C

flows influence the schedule feasibility of downstream flows, and vice versa, because both affect the time-windows of the vehicle. Infeasibilities or scheduling conflicts arise when a shipment k_1 on vehicle v tries to 'push' the time-windows to the latest allowed at some node i, while another shipment k_2 on the same vehicle v 'pulls' the time-windows at node i to be at the earliest allowed. We use *labels* to identify when shipments k_1 and k_2 give rise to a conflict at node i. To illustrate, consider the example in Table 4.3, with time-windows computed at node C of our example network. Note that shipment 2 must depart node C by 5, but shipment 3 cannot arrive at C before 5.2. Hence, there is no feasible schedule allowing shipments 2 and 3 to travel together from E to C, as required by the solution to the Flow Master Problem. We compute the time-window at C for shipments 2 and 3 as (5.2,5) = $(\max\{EAT_2^C, EAT_3^C\}, \min\{LDT_2^C, LDT_3^C\}) = (\max\{4.6, 5.2\}, \min\{5, 7\})$. This 'non-overlapping' time-window with negative duration, that is, $LDT < EAT$, represents an infeasible schedule. To eliminate associated infeasible shipment assignments, we add to the Flow Master Problem the following constraints:

$$f_{p_2} + f_{p_3} \leq 1. \tag{4.5}$$

This constraint enforces the restriction that shipments 2 and 3 cannot *both* be assigned as in the current solution to the Flow Master Problem.

After adding this constraint to the Flow Master Problem, we repeat, iterating between solving the Flow Master Problem and Scheduling Sub-Problem until a set of shipment paths and a feasible schedule for the shipments and vehicles is found; or the problem is shown to be infeasible.

4.2 Outline of the Algorithm

In this section, we more formally describe the algorithm, depicted in Figure 4-4, and extend it to the case under uncertainty. As described in Sections 3.2.2 and 3.2.3, uncertainty is captured using the ECCP in the Flow Master Problem and using the CCP in the Scheduling Sub-Problem. As observed in Section 4.1, we improve the solvability of the Flow Master Problem and Scheduling Sub-Problem by invoking a pre-processing step that identifies time-infeasible path assignments. The Pre-processing step is executed before the commencement of iterations solving the Flow Master Problem and Scheduling Sub-Problem. We use the same notation introduced in Chapter 3, and detail each module of the Decomposition algorithm in the following sections.

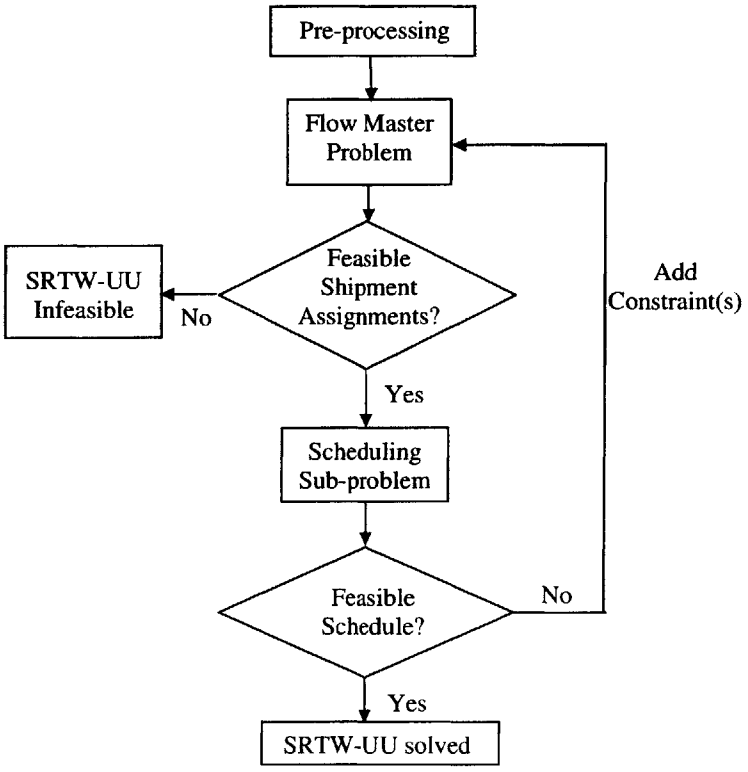

Figure 4-4: Flow Chart of the Decomposition Approach

We first present the notion of *shipment networks* and *aggregate networks*. *Shipment networks* $G_k = (N_k, A_k)$ for each shipment $k \in K$, are created as described in 3.1.1, such that initially $N_k = N$ and $A_k = \cup_{v \in V} A(v) \cup A'_k$. *Aggregate networks* are formed by superimposing the shipment and vehicle networks for all $k \in K$ over each other.

4.2.1 Step 1: Network Pre-processing

On G_k, for all $k \in K$, we define EAT_j^k as the earliest time shipment k can reach node j after starting from $O(k)$ and $sp_{O(k),j}^k$ is the shortest path distance for shipment k from $O(k)$ to node j. LDT_j^k is the latest time shipment k can leave node j to get to its destination $D(k)$ on time and $sp_{j,D(k)}^k$ as the shortest path distance for shipment k from node j to its destination. Similarly, EAT_j^v and LDT_j^v are the earliest arrival time and latest departure time of vehicle v at each node $j \in N$. In order to capture uncertainty in this module of the algorithm, we implement a CCP-based approach and decide a priori the quantiles of the service times (travel times, connection times and transfer times) for each arc based on the desired protection levels. These quantile values of service times are used in the computation of the shortest paths, the EATs and the LDTs.

The steps of the network Pre-processing phase are detailed as follows:

i For each shipment network G_k for all $k \in K$, based on the $EAT_{O(k)}^k$ and $LDT_{D(k)}^k$, we find time-windows, expressed in terms of the earliest arrival time EAT_i^k of each shipment $k \in K$ at any node $i \in N_k$ and the latest departure time LDT_i^k, of any shipment $k \in K$ at each node $i \in N_k$. EAT_i^k is the earliest time shipment k can reach node i (along a shortest path) from $O(k)$. LDT_i^k is the latest time at which shipment k can leave node i to reach D_k (along the shortest path) on time. Thus,

$$EAT_i^k = EAT_{O(k)}^k + sp_{O(k),i}^k \tag{4.6}$$

$$LDT_i^k = LDT_{D(k)}^k - sp_{j,D(k)}^k \tag{4.7}$$

ii We find time-windows (EAT_i^v, LDT_i^v) for each vehicle $v \in V$ at each node $i \in N$ along its route based on the earliest start time and latest return time required at the depot for the vehicle. These schedule constraints are driven by driver work rules. If vehicle v does not pass through a node i, the node is labeled unreachable for v. Thus, for all reachable arcs $(i,j) \in A_v$, for all $v \in V$, we have

$$EAT_j^v = EAT_i^v + tt_{ij}; \quad \text{and} \tag{4.8}$$

$$LDT_i^v = LDT_j^v - tt_{ij} \tag{4.9}$$

We refer to the time-windows of vehicles and shipments thus obtained as *pre-processing time windows.* They are the broadest set of time windows possible over any travel path for the vehicles and shipments (because these time-windows are formed using the *shortest* paths). These time-windows form the set of *pre-processing labels* at the nodes.

iii If the time-window duration of a shipment k is negative at a node $i \in N_k$, that is, $LDT_i^k - EAT_i^k < 0$, because the pre-processing time-windows are the broadest time-windows, no schedule feasible path for shipment k passes through i. For each shipment $k \in K$, we remove all the schedule infeasible nodes from G_k. We also remove all arcs in G_k incident to these nodes. This results in a reduced network $G_k = (N_k, A_k)$ containing only those arcs and nodes through which shipment k may pass.

iv We say that the time-windows of shipment k and vehicle v overlap at node i if the resulting *vehicle-shipment pre-processing time-window* has non-negative duration. Specifically, we denote the time-window at node i for the vehicle-shipment pair v and k as: $(EAT_i^{k,v}, LDT_i^{k,v})$, where $EAT_i^{k,v} = \max\{EAT_i^k, EAT_i^v\}$ and $LDT_i^{k,v} = \min\{LDT_i^k, LDT_i^v\}$. If the duration of this time-window is non-negative, that is, $LDT_i^{k,v} - EAT_i^{k,v} \geq 0$, then the time-windows of shipment k and vehicle v are said to overlap at node i. A non-negative duration implies

that it is possible for shipment k and vehicle v to be present at node i at the same time. We find all possible overlaps of shipment-vehicle pairs at each node $i \in N$. If the overlap between time-windows of shipment k and vehicle v is negative at node i, shipment k cannot travel on the vehicle arcs A_v incident to node i. We delete such arcs and nodes from the shipment network G_k, further reducing the size of G_k. If the time-windows of a vehicle-shipment pair (v, k) are non-zero at both ends of an arc $(i, j) \in A_v$ for any $v \in V$, shipment k can travel on (i, j) within the specified time-windows. We find all shipments $k \in K$ that can travel on an arc $(i, j) \in A_v$ by identifying all shipment-vehicle pairs with overlapping pre-processing time-windows at nodes i and j.

v Consider the aggregate network with the vehicle-shipment pre-processing time-windows for all $k \in K$ superimposed on each other at each node $i \in N$. Assume that shipments k_1 and k_2, each have overlapping feasible vehicle-shipment time-windows with vehicle $v \in V$ on arc $(i, j) \in A_v$. Assume that the time windows of the two shipment-vehicle pairs (v, k_1) and (v, k_2) at node i or node j do not overlap, thus making it infeasible for both k_1 and k_2 to travel together on $(i, j) \in A_v$ in any schedule-feasible solution to the SRTW-UU. This infeasibility can be eliminated from the set of feasible solutions to the Flow Master Problem by adding the following constraint to it:

$$\sum_{p_1 \in P_{k_1} | (i,j) \in p_1} f_{p_1} + \sum_{p_2 \in P_{k_2} | (i,j) \in p_2} f_{p_2} \leq 1, \tag{4.10}$$

where $f_p, p \in P_k$ is defined as before, that is, it is a binary variable that takes on value 1 if shipment k is assigned to path p; and 0 otherwise. We add these constraints to the Flow Master Problem in the Pre-processing step to eliminate known infeasible solutions.

4.2.2 Step 2: Flow Master Problem

In the nominal case, we formulate the Flow Master Problem as the SRTW-MP; and in the case with uncertainty in vehicle capacities, supplies and demands, we formulate the Flow Master Problem as the SRTW-UU-MP. These formulations, described in Section 3.2.2, are enhanced to include cuts generated in the Pre-processing stage and Scheduling Sub-Problem. The augmented SRTW-UU-MP is as shown below.

$$\max \quad \sum_{(i,j)\in A} w_{ij}\gamma_{ij} \tag{4.11}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{p\in P^k} d_k c_p f_p \leq \sum_{k\in K}\sum_{p\in P^k} d_k c_p f_p^{'*} + \Delta \tag{4.12}$$

$$\sum_{k\in K}\sum_{p\in P^k} d_k f_p \delta_{ij}^p \leq \sum_{l} u_{ij}^l (y_{ij}^l - y_{ij}^{l-1}) \qquad \forall\,(i,j)\in A, \forall\, l \in L_{ij} \tag{4.13}$$

$$y_{ij}^l \geq y_{ij}^{l-1} \qquad \forall\, l \in L_{ij} \tag{4.14}$$

$$y_{ij}^0 = 0 \qquad \forall\,(i,j)\in A \tag{4.15}$$

$$y_{ij}^L = 1 \qquad \forall\,(i,j)\in A \tag{4.16}$$

$$\sum_{p\in P^k} f_p = 1 \qquad \forall\, k \in K \tag{4.17}$$

$$\gamma_{ij} \leq \sum_{l=1}^{|L_{ij}|} p_{ij}^l (y_{ij}^l - y_{ij}^{l-1}) \qquad \forall\,(i,j)\in A \tag{4.18}$$

$$f_p \in \{0,1\} \qquad \forall\, p \in P^k, \forall\, k \in K \tag{4.19}$$

$$y_{ij}^l \in \{0,1\} \qquad \forall\, l \in L_{ij}, \forall\,(i,j)\in A \tag{4.20}$$

$$0 \leq \gamma_{ij} \leq 1 \qquad \forall\,(i,j)\in A \tag{4.21}$$

Cuts from the Pre-processing Step (4.22)

Cuts from the Scheduling Sub-Problem. (4.23)

4.2.3 Step 3: Scheduling Sub-problem (SRTW-UU-SP)

After solving the SRTW-UU-MP, vehicle routes and assigned shipment paths $p_1^{'}, ..., p_{|K|}^{'}$ are known. Though the paths introduced into the Flow Master Problem are individ-

ually schedule-feasible, it is still possible that interactions between these shipment paths $p_1^{'},...,p_{|K|}^{'}$ produce infeasible schedules. Therefore, in the Scheduling Sub-Problem, we 'flow' these shipments on the network G to determine if each of these movements is schedule feasible.

As in the case of the Pre-processing step, in solving the scheduling sub-problem for the nominal case, the travel times and connection time values are set to the nominal values. In order to capture uncertainty, again, we use the a priori quantiles used in the Pre-processing step. We provide more details on the a priori choices of the service time quantiles in Section 4.3.

The Scheduling Sub-Problem consists of the following steps on the aggregate network:

i Initialization: Let $k = 0$ represent the vehicle flows, and commodities $k = 1,...|K|$ represent the shipments.

(a) Set $EAT_i^k = 0$, and $LDT_i^k = M$, a very large number, for all $i \in N$, and for all $k \in K$

(b) Set $EAT_i = 0$, and $LDT_i = M$.

(c) Set processing list to empty.

(d) For $k = 0, 1, ...|K|$, determine the time-windows for commodity k along its currently assigned path $p_k^{'}$ in the aggregate network, as indicated in (4.24) and (4.25). That is, for each node $i \in N$, and $k = 0, 1, ..., |K|$:

$$EAT_j^k = \max EAT_j^k, EAT_i^k + tt_{ij} \quad \forall\,(i,j) \in p_k^{'}, \forall\, k = 0, 1, ..., |K| \quad (4.24)$$

$$LDT_i^k = \min LDT_i^k, LDT_j^k - tt_{ij} \quad \forall\,(i,j) \in p_k^{'}, \forall\, k = 0, 1, ..., |K| \quad (4.25)$$

These time-windows will at least be as tight as the Pre-processing time-windows obtained in Step 1, because each shipment $k \in K$ is restricted

to path p_k'. The time-windows for the vehicles ($k = 0$) remain the same as those in the Pre-processing step, because the vehicle routes are given inputs that do not change in solving the SRTW-UU.

ii For node $i \in N$ and $k = 0, 1, ..., |K|$,

(a) if $EAT_i^k \neq EAT_i$ and $EAT_i^k \neq 0$ set $EAT_i^k = EAT_i$,

(b) if $LDT_i^k \neq LDT_i$ and $LDT_i^k \neq M$ set $LDT_i^k = LDT_i$,

(c) and add k to the processing list, if it not already present.

We refer to (EAT_i, LDT_i) as the *movement time windows* at node i.

iii If the processing list is not empty, remove the first element from the list, and go to step iv, else go to step v. If the processing list is empty, it indicates that the widest movements time-windows allowing all the flows that are output from the Flow Master Problem, are found.

iv Update EAT_i^k, LDT_i^k for all $(i,j) \in p_k'$, processing (i,j) in sequence along p_k' as:

$$EAT_j^k = \max\{EAT_j^k, EAT_i^k + tt_{ij}\} \tag{4.26}$$

$$LDT_j^k = \min\{LDT_j^k, LDT_i^k - tt_{ij}\} \tag{4.27}$$

Return to Step iii.

v Define $l_{ij} = LDT_j - EAT_i, \forall (i,j) \in A$. Find the shortest, non-cyclic path p^*, beginning and ending at any node, in the aggregate network with arc lengths l_{ij}. Suppose the length of p^* is l^*. If path p^* has negative length, that is, $l^* < 0$, let K^* be the set of commodities assigned to at least one arc in p^*; otherwise stop - a feasible schedule has been identified.

vi Add the following constraint to the Flow Master Problem:

$$\sum_{k \in K^*} f_{p_k'} \leq |K^*| - 1; \tag{4.28}$$

with $|K^*|$ representing the size of the set K^*. Constraint 4.28 states that the set of paths causing schedule infeasibility of the current solution should not be repeated in further iterations.

4.2.4 Stopping Criterion

If no infeasibilities are identified in the scheduling algorithm, that is, $l^* \geq 0$ the SRTW-UU is solved; otherwise, the algorithm returns to Step 2 with added constraints of the type (4.28) in the Flow Master Problem. We iterate between solving the Flow Master Problem and the Scheduling Sub-Problem until either the algorithm terminates with a feasible schedule for the Flow Master Problem solution, in which case we have found the optimal solution to the original problem; or if the Flow Master Problem is determined to be infeasible, in which case, the SRTW is infeasible.

4.2.5 Output

The solution obtained from the above algorithm is a set of paths to which shipments are assigned and a set of *time-windows* indicating the earliest and latest time each vehicle and shipment movement can occur. These time-windows provide bounds within which the current set of vehicle and shipment flows may be scheduled. In the case when schedule delays and disruptions occur, the windows allow the solution to remain feasible at times, even when re-scheduling is necessary. This is indicative of the *inherent flexibility* provided by the solution to our decomposition approach.

4.3 Comments on the Robust Decomposition Approach

4.3.1 Elimination of Infeasible Space Using Cuts

The correctness of the decomposition approach is dependent on the fact that the cuts introduced in the Pre-processing and Scheduling Sub-Problems eliminate that

region of the Flow Master Problem solution space that is infeasible to the original SRTW(-UU) problem, and ensure that the current infeasible solution is not repeated.

Proposition: The cuts generated in the Pre-processing module and Scheduling Sub-Problem correspond to infeasible SRTW(-UU) solutions, and do not eliminate any feasible SRTW(-UU) solutions.

Proof: In the Pre-processing stage, the pre-processing time-windows identified are the broadest possible time-windows for the shipment and vehicle movements because the shortest path is used in determining these windows. Therefore, when we eliminate nodes and arcs from the network, we eliminate those solutions that cannot satisfy schedule constraints under any conditions. Similarly, when we generate constraints of the type (4.10), we are eliminating all vehicle-shipment pairs that are schedule-infeasible even under the broadest time-windows. Hence, such vehicle-shipment pairs cannot travel together in *any* solution. Thus, constraints (4.10) are valid, and do not eliminate more feasible space from the Flow Master Problem than necessary.

In the case of the Scheduling Sub-Problem, the movement time-windows on the aggregate network are the broadest possible time-windows for the shipments and vehicles as assigned in the Flow Master Problem solution. If all nodes and arcs have scheduling time-windows that are non-negative, then one or more feasible schedules can be constructed. One trivial case is to set the scheduled time at each node $i \in N$ to EAT_i. Alternatively, a feasible schedule can be constructed by setting the scheduled time at each node $i \in N$ to LDT_i. If we find instead, that some of the movement time-windows are of negative duration, that is, $l^* < 0$, there does not exist a feasible schedule for the current Flow Master Problem solution. $l^* < 0$ for path p^* implies that $\sum_{(i,j)\in p^*} tt_{ij} > LDT_e - EAT_s$, with s and e representing the start and end nodes, respectively, of path p^*. Hence, the minimum time necessary to traverse the arcs in p^* exceeds the maximum allowable time to get from s to e. Constraints of the form (4.28) eliminate only the portion of the current solution creating the schedule infeasibility. One can add more constraints of the form (4.28) by running additional shortest path algorithms from different nodes in the aggregate networks and identifying other paths

with negative length, if they exist.

4.3.2 Convergence of the Algorithm

In each iteration of the decomposition approach, we solve a relaxed version of the formulation (3.9) - (3.16). Thus the cost incurred by the solution to the SRTW(-UU)-MP is a lower bound on the objective function cost of the SRTW. As we add cuts to the Flow Master Problem, the objective function cost increases (or stays the same) because we are further constraining the minimization problem. Each cut corresponds to the elimination of at least one infeasible solution to the SRTW(-UU). Hence, each cut is unique, and after a finite (but possibly large) number of iterations, all infeasible solutions are eliminated and the Flow Master Problem solution will be feasible to the SRTW(-UU). Because the Flow Master Problem is a relaxation of SRTW(-UU) and the added cuts eliminate only infeasible SRTW(-UU) solutions, finding a feasible schedule to the Flow Master Problem corresponds to solving, that is, finding an optimal solution to, the SRTW(-UU).

4.3.3 Identification of Dominant Cuts

We can strengthen the constraints in the Pre-processing and Scheduling Sub-Problem as described below.

Consider constraints in the Scheduling Sub-Problem of the form:

$$f_{p_1} + f_{p_2} \leq 1, \tag{4.29}$$

$$f_{p_1} + f_{p_3} \leq 1, \quad \text{and} \tag{4.30}$$

$$f_{p_2} + f_{p_3} \leq 1. \tag{4.31}$$

Notice that these three constraints can be modeled effectively with a single, *dominant* constraint, of the form:

$$f_{p_1} + f_{p_2} + f_{p_3} \leq 1. \tag{4.32}$$

Similarly, in the Pre-processing step, suppose shipments k_1, k_2 and k_3 are identified, such that each pair cannot travel together on an arc (i,j), pair-wise constraints of type (4.10) are generated for k_1 and k_2, k_2 and k_3, k_3 and k_1. These constraints can be replaced by a single *dominant constraint*:

$$\sum_{p_1 \in P_{k_1} | (i,j) \in p_1} f_{p_1} + \sum_{p_2 \in P_{k_2} | (i,j) \in p_2} f_{p_2} + \sum_{p_3 \in P_{k_3} | (i,j) \in p_3} f_{p_3} \leq 1. \tag{4.33}$$

Identifying such dominant constraints becomes important when we have a large number of shipments and vehicles, as it reduces the size of the Flow Master Problem and it can reduce the number of iterations of the algorithm necessary to solve the SRTW-UU. Dominant constraints can be identified by finding *maximal completely connected components* (called *cliques*) in a particularly constructed graph. We discuss clique algorithms in Section 4.4.

4.3.4 Capturing Service Time Uncertainty

To capture uncertainty in service times, we use a priori defined quantile values of service time (travel time, transfer time, connection time) on each arc. These quantile values, representing the protection levels we want to provide, provide additional slack in the service times, thereby acting as 'buffers'. The quantile value for the *arc* service time can be picked based on the desired protection level for the *path*. For example, to protect against uncertainty in service times to the 90% level, we can determine the appropriate value of the service time on each arc, making assumptions such as independence of arc service times. With dependent service times, we can use historical path-based service times and set arc-based quantile values to achieve the desired levels of protection approximately.

Consider a shipment path $ij_1, ij_2, ..., ij_n$ containing n arcs. Let t_{ij} for all $(i,j) \in A$ be the service time on arc (i,j). If t_{ij} for all $(i,j) \in A$ are independent random variables, and T_{ij} for all $(i,j) \in A$ represent, say, the 90% quantile values of the arc service times, then $\Pr(t_{ij_p} < T_{ij_p}) = 0.9$, for all $p = 1, ..., n$, and:

$$\Pr(t_{ij_1} < T_{ij_1}, ..., t_{ij_f} < T_{ij_f}, ...t_{ij_n} < T_{ij_n}) = \prod_{p=1}^{n} Pr(t_{ij_p} < T_{ij_p}) = 0.9^n, \text{ and} \quad (4.34)$$

$$\Pr(\sum_{p=1}^{n} t_{ij_p} > \sum_{p=1}^{n} T_{ij_p}) <= 1 - 0.9^n. \quad (4.35)$$

This provides a weak bound on the level of protection of the service time along the path $ij_1, ij_2, ..., ij_n$.

Stronger bounds can be obtained for some simple cases. For example, if the t_{ij} are independent uniform [0,1] random variables and all the T_{ij} are equal to α, where $\frac{n-1}{n} \leq \alpha \leq 1$, then

$$\Pr(\sum_{p=1}^{n} t_{ij_p} > \sum_{p=1}^{n} T_{ij_p}) = \Pr(\sum_{p=1}^{n} t_{ij_p} > n\alpha) = \frac{(n(1-\alpha))^n}{n!}. \quad (4.36)$$

This result is easily generalized to a more 'realistic' scenario, e.g., all the t_{ij} are uniform [a,b] random variables and all the T_{ij} are scaled appropriately. Similarly, strong bounds exist for other special cases such as uncorrelated Gaussian random variables, and independent and identically distributed exponential random variables. These assumptions are not necessarily overly restrictive, as travel time distributions have historically been well-approximated by some of these special cases. Therefore, the most appropriate quantiles to be adopted as a priori protection levels in the Pre-processing and the Scheduling Sub-Problems can be determined through the use of these bounds.

4.3.5 Running Time of the Algorithm

Pre-processing involves employing network labeling and shortest path algorithms. We solve $O(2(|K|+|V|))$ shortest path problems, for which highly efficient algorithms are available. We note that in the Pre-processing stage, the shipment networks are reduced in size, due to elimination of arcs and nodes. Computation of time-windows

involves computation at each node and arc of each shipment network, requiring a maximum of $|K|(|N| + |A|)$ computations.

The Flow Master Problem is solved by standard integer programming techniques. It is more tractable than conventional SRTW formulations, however, its size is expected to be smaller, even with the addition of cuts from the Pre-processing and Scheduling Sub-Problems, than the size of the formulation of (3.1) - (3.8).

The Scheduling Sub-Problem involves network labeling algorithms to compute the time-windows, similar to the Pre-processing stage. Tracing the paths of commodities involves $O(|N|^2(|K| + |V|) + |N|(|K| + |V|))$ operations, because label initialization requires $O(|N|(|K| + |V|))$ and each time a shipment or vehicle path is traced, one label at some node i is set to EAT_i or LDT_i, and it will not be changed again in the algorithm. The number of labels at each node is restricted to $2(|K| + |V|)$ and each relabeling tkes $O(|N|)$ steps.

4.3.6 Advantages of the Decomposition Approach

We summarize the advantages of our decomposition approach as:

- The decomposition methodology involves solving a multi-commodity Flow Master Problem and a series of network-based, easy-to-solve sub-problems. Because we are breaking a large optimization problem into two smaller parts, one involving finding an optimal solution and the other simply finding a feasible solution, tractability is enhanced.

- The Flow Master Problem is a simple multi-commodity flow problem, for which standard solution techniques exist.

- Capturing uncertainty in the decomposition approach does not incur much additional complexity relative to solving the SRTW for the nominal case.

- Specific types of additional constraints, such as those restricting time-windows of a vehicle at a particular location, can be added without increasing algorithmic complexity.

4.3.7 Limitations of the Decomposition Approach

We summarize the limitations of our decomposition approach as follows:

- Correlations between uncertain parameters are not captured. For example, it is possible that travel times on adjacent arcs on the network are correlated. As yet, we cannot capture such correlations.
- The choice of 'protection levels' for travel times on the network has to be made a priori, and involves repeated solution of instances of our model. As in the case of applying the extended CCP to the Flow Master Problem, it would be useful to develop a mechanism to automate the choice of protection levels.

4.4 Using Cliques to Find Dominant Cuts

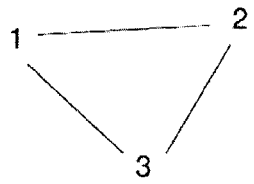

Figure 4-5: Incompatibility Network: Cliques

As introduced in Section 4.3.3, in the Pre-processing step, after considering all possible pairs of shipments on the aggregate network, we can replace weak cuts of the

form (4.10) by stronger constraints as described in (4.33). Similarly, in the Scheduling Sub-Problem, we can identify constraints, for example, (4.29) - (4.31) that can be replaced by stronger constraints, such as $f_{p_1} + f_{p_2} + f_{p_3} \leq 1$. One approach to finding such constraints is to construct an *incompatibility* network over which completely connected subgraphs, called cliques, are identified. To construct the incompatibility network, we create one node for each path in the Flow Master Problem solution. An arc connects a pair of nodes if the associated paths are contained in at least one constraint that is added to the Flow Master Problem as a result of the Pre-processing or Scheduling Sub-Problem solution steps. Each completely connected subgraph in the incompatibility network is a clique that corresponds to set of paths (the nodes of the clique), of which at most one can exist in a solution. We find dominant, or strong cuts, by identifying cliques of maximal size, that is, with the largest number of nodes. For the set of constraints (4.29) - (4.31), Figure 4-5 is the incompatibility network, giving rise to the dominant constraint (4.32).

Identification of dominant cuts minimizes the number of cuts that must be identified and added to the Flow Master Problem, thus potentially reducing the number of iterations of the decomposition algorithm and leading to faster solution times. Note that identifying cliques and stronger constraints requires additional computation time. It is necessary, then, to find appropriate trade-offs between increased time to identify stronger constraints and the corresponding reduction in overall solution time.

We describe a basic algorithm that finds cliques, in the following section.

4.4.1 Clique Algorithms

Tarjan [32] presents one of the earliest and best algorithms to find cliques in a graph. Here we present a version of Tarjan's algorithm. Consider a directed graph $G = (V, E)$, where V is the set of nodes and E the set of edges of the graph. Tarjan has presented an algorithm that finds the cliques in $O(|V| + \varepsilon)$ time. The algorithm is presented below:

```
procedure VISIT(v)
begin
    root[v] = v, InComponent[v] = false;
    PUSH(v, stack);
    for each node w such that (v, w) ∈ E do begin
        if w is not visited then VISIT(w);
        if not InComponent[w] then root[v] = min{(root[v], root[w])};
    end;
    if root[v] = v then
        repeat
            w = POP(stack);
            InComponent[w] = true;
        until w = v;
end;

begin /* Main program */
stack = φ;
for each node v ∈ V do
    if v is not already visited then VISIT(v);
end;
```

Tarjan's algorithm is asymptotically efficient. More efficient algorithms that build upon the above have been proposed in Nuutila and Soisalon-Soininen [26] and Wood [33]. Though the problem of finding cliques in a network is NP-hard, because we expect to have only a subset of shipments incurring infeasibilities at a particular node, we expect that the size of our incompatibility network will be small and therefore tractability should not be an issue. Also, we need not identify all cliques in the graph to improve the algorithmic efficiency; adding even a few clique constraints can improve algorithmic performance.

4.5 Applicability to Large-scale Problems

In large-scale SRTW problems, each shipment will have a number of associated paths to which it can be assigned and, likely, a very large number of possible schedules. Enumerating all path-schedule combinations for all shipments is often impractical, thereby leading to issues of intractability. In this section, we discuss how our decomposition approach is well-structured to solve large-scale problems.

Because we strip out schedule considerations, and use a static network in the decomposition algorithm, the number of network paths for each shipment is likely to be manageable, even if many paths exist for each commodity. The size of the Flow Master Problem, is therefore, much smaller than that of (3.9) - (3.16) or (3.17) - (3.22). Including a variable corresponding to each path in the Flow Master Problem might be tractable for some instances. And, in those cases for which intractability is encountered due to the presence of all paths in the formulation of the Flow Master Problem, we can resort to explicit column generation. This involves enumerating all the paths in the network and placing them in a 'pool'. Only a limited number of paths from this pool are initially included in the Flow Master Problem formulation. At each iteration, we explicitly price-out all the paths in the 'pool' and add all or a subset of paths that have negative reduced-costs to the Flow Master Problem formulation. This is likely to be practical, even for large problem instances.

Note that in the Pre-processing stage, we eliminate paths that are unreachable or schedule-infeasible (Section 4.3), thus, reducing the number of paths in the 'pool'. An important observation in this regard is that inclusion of schedule decisions in the problem to be solved using conventional approaches causes formulation size to *increase*, whereas their inclusion results in a decrease in the size of the SRTW(-UU)-MP.

Note that when solving the problem in the nominal case or under uncertainty, the structure of SRTW(-UU)-MP allows the use of multi-commodity flow solution

techniques.

4.6 Illustration 2

We apply our decomposition approach and other conventional approaches to a scaled-down version of a SRTW-UU problem. The problem instance contains 5 locations, 3 vehicles and 6 shipments. Each vehicle has a capacity of 2 units. Each arc has a travel time of 1 unit and connections and transfers take 0.2 units in the nominal case. Costs are associated with non-service of shipments.

In this example, we deal with uncertainty in service times, and in particular, we assume uncertainty to exist only in the connection times. Note that this assumption incurs very little loss of generality because uncertainty in travel times can similarly be captured by 'shifting' the uncertainty onto the connection times. We assume that the connection times all vary uniformly between values of 0 and 0.4 time units. The value used for the nominal case is the commonly used expected value, that is, 0.2.

We solve this problem both with and without uncertainty. In the nominal case, the problem is solved using the conventional formulation of (3.1) - (3.8) and our decomposition approach. In the robust case, we solve it using the Bertsimas and Sim approach and our decomposition approach. In the Bertsimas and Sim approach, each constraint has just one uncertain parameter per constraint, and because its protection parameter is usually an integer, it uses the worst-case value of 0.4 for connection times. In the robust decomposition approach, we choose a protection level of 75%, yielding a value of 0.3 for the service time.

The solutions obtained from each approach were tested for feasibility by simulating randomly generated scenarios, with the realization of each of 100 scenarios constructed by sampling from a uniform distribution between 0 and 0.4. The results from the approaches are summarized in Table 4.4.

	Nominal solutions		Robust solutions	
	Static	Decomposition	Bertsimas-Sim	Decomposition
Solution Nature	Exact times	Time-windows	Exact times	Time-windows
Cost	0	0	1 shipment not serviced	0
% scenarios feasible	18	73	100	95

Table 4.4: Computational Results of the Decomposition Approach on a Small Instance

For the nominal case, we observed that the optimal schedule from the conventional approach is contained within the time-windows obtained by our decomposition approach, that is, they formed a proper subset of all solutions. Our decomposition approach, however, outputs all possible solutions. Because the solution to the decomposition approach is in terms of time-windows, it indicates the extent to which the schedule can be changed while keeping the current solution feasible, and thus it is feasible for 73% of the sampled scenarios. In contrast, the conventional approach, with its exact output schedule, was feasible for only 18% of the sampled scenarios.

When incorporating uncertainty, the Bertsimas-Sim approach, due to its higher protection level, results in a solution in which one shipment is not delivered. The solution, however, is feasible for 100% of the scenarios. The solution corresponds to an 83.3% (5/6*100) average service level. The decomposition approach, however, serves all the shipments for 95% of the scenarios. The Bertsimas-Sim approach requires no re-planning for any scenario (it is always feasible) while the decomposition approach required replanning 5% of the time. The two approaches, therefore, present solutions that trade-off re-planning requirements and service level.

We understand, from this illustration, that metrics of robustness may often be conflicting, and a higher level of protection need not necessarily perform 'better' with respect to all metrics. Therefore, there is a necessity to define robustness for each problem, based on the needs of the user, and tailor the robust solution approach to the relevant metrics.

4.7 Proof of Concept

In this section, we apply our decomposition approach to a large-scale problem involving dynamic airline scheduling. We compare its performance with conventional models that are built using the time-space network approach discussed in Section 3.1.2.

Dynamic scheduling addresses the problem of demand stochasticity faced by airlines. Because airlines determine their flight schedules a year to six months in advance, when demand forecasts are highly uncertain, they face the issue of matching capacity to demand during operations. Demand forecasts for flights become more accurate as the date of flight approaches, giving the airline an opportunity to match capacity to demand better by adopting a dynamic scheduling approach.

Traditionally, dynamic scheduling consisted of flight re-fleeting alone. Jiang [21] introduces fleet re-timing as a dynamic scheduling mechanism and supplements re-fleeting with re-timing. Re-timing the schedule and re-fleeting of aircraft increase or decrease the number of connecting itineraries available to passengers (compared to the original schedule) and increase or decrease the number of seats available in the affected markets. This can help to reduce passenger spill by better matching capacity. The dynamic scheduling approach modifies the existing flight schedule and fleet assignments, keeping existing bookings still feasible (though possibly re-timed), so that *realized* demand can be accommodated as much as possible. Jiang [21] shows that through the dynamic mechanisms of flight re-timing and re-fleeting, even 'optimized' schedules can be improved by re-designing the schedule at regular intervals.

By performing dynamic scheduling during the booking period, the airline can supplement revenue management techniques that manage the demand side, with re-fleeting and re-timing the supply side. Especially in the case of de-banked hubs, minor adjustments to flight leg arrival and/or departure times can affect the set of

connecting itineraries served through that hub. In fact, flight schedule re-timings can increase or decrease the supply of available seats in markets connecting at the hub [21]. For each daily schedule available, Jiang sets the period of re-optimization of schedule to be 21 days before departure of the flight. Further details of Jiang's modeling approach are available in Jiang [21].

4.7.1 Traditional Solution Approaches

In this section, we examine in detail the approach proposed by Jiang for dynamic scheduling. His network representations and model formulations follow.

Network Description

To achieve schedule re-optimization, two tailored time-space networks are created. One is a passenger flow network and the other a set of aircraft flow networks, created for each fleet type. The flow of aircraft over the airline fleet schedule, for each fleet type k, is modeled in the aircraft flow network G_k. In the network G_k, every node corresponds to departure time of a flight leg f in the flight schedule, or to the arrival time plus minimum turn time for fleet type k at the arrival location of flight leg f. An arc in G_k is either a ground arc or a flight arc, with flight arcs representing flights in the schedule, and ground arcs representing an aircraft waiting on the ground at the same location. Wrap-around arcs are ground arcs that connect the first and last node at every location, representing balance of aircraft. We also define a count line, which is arbitarily defined at some point in time, and used to count the number of aircraft of fleet type k used on network G_k, to operate the flight schedule.

The passenger flow network G models the passenger flow over the network G, which is fleeted. In this network, each node is either the departure time of a flight leg at the corresponding location, or the arrival time of a flight leg at the location. Arcs are classified as flight arcs and connection arcs. Flight arcs represent flights in the current schedule. A connection arc is created to connect two flight arcs if a passenger can connect between the two corresponding flights, that is, minimum connection time and

maximum connection time restrictions are satisfied. A path in this network represents a feasible itinerary for a passenger. Capacities of the flight arcs are determined by the fleet type assigned to the flight leg represented by that arc.

Problem Formulation

In order to capture the idea of re-timing of flights, copies of each flight leg are made at discrete intervals on the passenger and aircraft flow networks. Among all copies of each flight leg, only one has to be in the solution, resulting in the new re-timed schedule. This involves making decisions about the presence or absence of arcs in the network, making the problem one of network design, as defined in Chapter 1. The notation and formulation presented by Jiang [21] is as follows.

Notation:

- Π: set of fleet types.
- S: set of cities.
- G^{π}: set of ground arcs in fleet $\pi \in \Pi$'s network.
- $FML(\pi)$: set of fleet types in the same family as $\pi \in \Pi$.
- D_m^F: demand forecast for market m.
- $fare_m^F$: forecasted average fare for demand in market m.
- $y_{\pi}^{i^0}$: number of aircraft overnighted at city $i \in S$ for fleet type π in the original schedule.
- π_l^0: the fleet type used on leg $l \in L$ in the original schedule.
- T: set of time intervals at the hub, indexed by t.
- MAX^{at}: maximum number of aircraft arrivals at the hub in the interval $t \in T$.
- MAX^{dt}: maximum number of aircraft departures from the hub in the interval $t \in T$.

- $\zeta^i_{g\pi} = \begin{cases} 1, \text{ if arc } g \in G^\pi \text{ is a wrap around arc at city } i \in S; \\ 0, \text{otherwise} \end{cases}$

- $C(l)$: set of flight copies for flight leg $l \in L$.

- $\langle l, k \rangle$: copy $k \in C(l)$ of flight leg $l \in L$.

- $c_{lk\pi}$: cost to fly $\langle l, k \rangle$ with aircraft type $\pi \in \Pi$, where $k \in C(l), l \in L$.

- c_π: fixed cost to have one additional aircraft type $\pi \in Pi$.

- N^π: nodes in flight network of fleet type $\pi \in \Pi$.

- n_π: number of aircraft available for fleet type $\pi \in \Pi$.

- BKD_l: number of seats already booked on flight leg $l \in L$ before re-optimization.

- $\bar{\alpha}^i_{lk\pi} = \begin{cases} 1, \text{ if } \langle l, k \rangle \text{ in fleet } \pi\text{'s network begins at node } i \in N^\pi; \\ -1, \text{ if } \langle l, k \rangle \text{ in fleet } \pi\text{'s network terminates at node } i \in N^\pi; \\ 0, \text{ otherwise.} \end{cases}$

- $\hat{\alpha}^i_{g\pi} = \begin{cases} 1, \text{ if ground arc } g \in G^\pi \text{ begins at node } i \in N^\pi; \\ -1, \text{ if ground arc } g \in G^\pi \text{ terminates at node } i \in N^\pi; \\ 0, \text{ otherwise.} \end{cases}$

- $\bar{\beta}_{lk\pi} = \begin{cases} 1, \text{ if c in fleet } \pi\text{'s network crosses the count line;} \\ 0, \text{ otherwise.} \end{cases}$

- $\hat{\beta}_{g\pi} = \begin{cases} 1, \text{ if ground arc } g \in G^\pi \text{ crosses the count line;} \\ 0, \text{ otherwise.} \end{cases}$

- $\gamma^{at}_{lk} = \begin{cases} 1, \text{ if } \langle l, k \rangle \text{ arrives at the hub during interval } t \in T; \\ 0, \text{ otherwise.} \end{cases}$

- $\gamma^{dt}_{lk} = \begin{cases} 1, \text{ if } \langle l, k \rangle \text{ departs from the hub during interval } t \in T; \\ 0, \text{ otherwise.} \end{cases}$

- Q: set of connecting itineraries that are booked previously.

Decision Variables:

- $f_{lk\pi} = \begin{cases} 1, & \text{if fleet } \pi \in \Pi \text{ is used to fly flight copy } \langle l,k \rangle, \text{ where } k \in C(l), l \in L; \\ 0, & \text{otherwise.} \end{cases}$

- $y_{g\pi}$: number of aircraft on ground arc $g \in G^\pi$.

- z_π: number of aircraft used for fleet type π.

Formulation:

$$\max \quad \sum_{m\in M}\sum_{r\in R(m)} x_{mr} fare^F_m - \sum_{l\in L}\sum_{k\in C(l)}\sum_{\pi\in\Pi} c_{lk\pi} f_{lk\pi} - \sum_{\pi\in\Pi} z_\pi c_\pi \tag{4.37}$$

$$\sum_{k\in C(l)}\sum_{\pi\in\Pi} f_{lk\pi} = 1, \forall l \in L \tag{4.38}$$

$$\sum_{r\in R(m)} x_{mr} \le D^F_m, \forall m \in M \tag{4.39}$$

$$\sum_{m\in M}\sum_{r\in R(m)} \delta^{lk}_{mr} x_{mr} \le \sum_{\pi\in\Pi} f_{lk\pi}(CAP_\pi - BKD_\pi), \forall l \in L, k \in C(l) \tag{4.40}$$

$$\sum_{l\in L}\sum_{k\in C(l)} f_{lk\pi}\bar{\alpha}^i_{lk\pi} + \sum_{g\in G^\pi} y_{g\pi}\hat{\alpha}^i_{g\pi} = 0, \forall i \in N^\pi, \pi \in \Pi \tag{4.41}$$

$$\sum_{l\in L}\sum_{k\in C(l)} f_{lk\pi}\bar{\beta}_{lk\pi} + \sum_{g\in G^\pi} y_{g\pi}\hat{\beta}_{g\pi} = z_\pi, \forall \pi \in \Pi \tag{4.42}$$

$$z_\pi \le n^\pi, \forall \pi \in \Pi \tag{4.43}$$

$$\sum_{l\in L}\sum_{k\in C(l)} \gamma^{at}_{lk} \sum_{\pi\in\Pi} f_{lk\pi} \le MAX^{at}, \forall t \in T \tag{4.44}$$

$$\sum_{l\in L}\sum_{k\in C(l)} \gamma^{at}_{lk} \sum_{\pi\in\Pi} f_{lk\pi} \le MAX^{dt}, \forall t \in T \tag{4.45}$$

$$f_{lk\pi\prime} = 0, \forall l \in L, k \in C(L), \pi\prime \notin FML(\pi^0_l) \tag{4.46}$$

$$\sum_{g\in G^\pi} y_{g\pi}\zeta^i_{g\pi} \le y^{i^0}_\pi, \forall i \in S, \forall \pi \in \Pi \tag{4.47}$$

$$\sum_{\pi\in\Pi} f_{l_1k_1\pi} + \sum_{\pi\in\Pi} f_{l_2k_2\pi} \le 1, \forall (l_1,l_2) \in Q, (k_1,k_2) \notin C(l_1,l_2) \tag{4.48}$$

$$f_{lk\pi} \in \{0,1\}, \forall l \in L, k \in C(l), \pi \in \Pi \tag{4.49}$$

$$x_{mr} \geq 0, \forall m \in M, r \in R(m) \tag{4.50}$$

$$y_{g\pi} \geq 0, \forall g \in G^{\pi}, \pi \in \Pi \tag{4.51}$$

$$z_{\pi} \geq 0, \forall \pi \in \Pi \tag{4.52}$$

Constraints (4.38) enforce that each flight leg is covered exactly once. Constraints (4.39) limit the number of passengers traveling in each market to the value of that market's unconstrained demand. Constraints (4.40) do not allow the number of future passenger bookings to exceed the number of remaining available seats. Constraints (4.41) ensure flow balance of aircraft. Constraints (4.42) and (4.43) ensure that the re-optimized schedule uses no more aircraft than the original schedule. (4.44) and (4.45) constrain the number of departure and arrival activities in each time interval to the maximum number of allowable activities. Constraints (4.46) enable re-fleeting within fleet families. Constraints (4.47) position aircraft as in the original schedule, at the beginning and end of each day. Constraints (4.48) ensure service to passengers whose itineraries are previously booked. Constraints (4.49), (4.50), (4.51) and (4.52) enforce the ranges of possible variable values.

4.7.2 Decomposition Solution Approach

In the approach presented in Section 4.7.1, copies of arcs are made in order to indicate the flexibility in the schedule, and only one copy of each flight leg is picked in the solution, indicating the optimal schedule. We now present a decomposition solution algorithm for dynamic scheduling, a network design problem.

Step 1: Initialization

We create passenger flow networks G' and aircraft flow networks G'_k, for all fleet types k, in which no copies are made. We provide the broadest possible schedules to the Flow Master Problem passenger flow networks and aircraft flow networks as follows: in the passenger flow network, if a connection is possible between flight leg l_1 and flight leg l_2 by means of re-timing (if copies were used, as in Section 4.7.1,

this would be interpreted as having a connection between some copy k_1 of flight leg l_1 and some copy k_2 of flight leg l_2, obeying the minimum connection time $MinCT$ and maximum connection time $MaxCT$ constraints), then we allow flight legs l_1 and l_2 in the decomposition network to be connected by means of connection arcs in the passenger flow networks. In the aircraft flow networks G'_k, because minimum turn times are incorporated directly in the flight arcs, any pair of flight legs having an ordinal relationship (one arc follows the other in time) can be connected by means of fleet connection arcs. This design of the Flow Master Problem network allows passenger and aircraft paths with all possible schedules for the aircraft and passengers to be represented by means of the reduced network.

Figure 4-6 shows a connection in the time-space network G'. Copies for the flight arcs are not created. Normally, the flight legs (represented by the bold lines) cannot connect; however, connection arcs are added between the two flight legs because if re-timing is allowed, flight leg l_1 can connect with flight leg l_2 without exceeding the $MaxCT$.

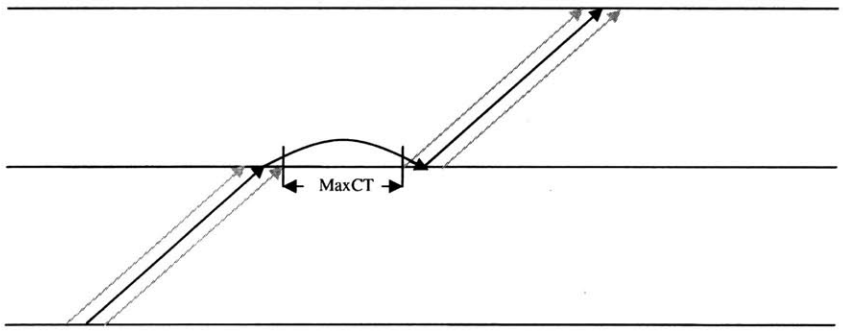

Figure 4-6: Connections in the Passenger Flow Network G'

Step 2: Flow Master Problem

The Flow Master Problem solution is an assignment of aircraft to flight legs, as well as passengers to itineraries, such that the total profit, defined as the difference between the revenue and cost, is minimized, by solving (4.37) - (4.52) on the underlying

networks G and G_k for all fleet types k.

The Flow Master Problem solution is an assignment of itineraries to the passengers on the passenger flow network, from passenger origins to their respective destinations. It is also an assignment of fleet types to flight legs, satisfying fleet count constraints.

Because we are solving a relaxed version of the original problem, the solution objective function value of this Flow Master Problem is an upper bound on the value of the objective function of the dynamic scheduling problem.

Step 3: Scheduling Sub-Problem

Because no constraints pertaining to schedule feasibility are included in the Flow Master Problem and the network considered is not expanded using copies to capture exact schedules, after the Flow Master Problem is solved, the passenger paths or itineraries in the solution can be infeasible. This can happen because the passenger $MinCT$ and $MaxCT$ are not considered explicitly in the construction of the network G', and therefore, the passenger itineraries obtained as output from the Flow Master Problem might not satisfy the $MinCT$ and $MaxCT$ restrictions. Therefore, in the Scheduling Sub-Problem, we examine the solution from the Flow Master Problem, and check if it has a feasible schedule. Note that in this solution, all aircraft have a feasible schedule, because fleet networks G'_k incorporate fleet minimum turn times, and therefore, any set of flight arcs that can be connected by means of ground arcs has a feasible schedule.

In the Scheduling Sub-Problem, therefore, we examine each of the passenger itineraries, and check if the connections in the itinerary satisfy the $MinCT$ and $MaxCT$ constraints. If a passenger itinerary r in market m is infeasible, then re-timing of some of the flights constituting the itinerary might be required, or the passenger might be transferred to some other itinerary in the same origin-destination market.

In order to eliminate the infeasibility, we re-formulate the networks G' and G'_k for all k, with copies made for all flight legs present in each itinerary r in market m with schedule infeasibility, and create connection arcs on the passenger flow network and aircraft flow networks for the new set of arcs that satisfy exactly the $MinCT$ and $MaxCT$ conditions. This will allow us to capture all possible schedules for the passenger, eliminating infeasibilities present in the current solution to the Flow Master Problem. With this modified network as the underlying network, we again go to Step 2, and iterate.

Stopping Criterion

Different stopping criteria such as the number of iterations, the number of passengers with infeasible paths, etc., can be defined by the user.

If a feasible solution to the problem (4.37) - (4.52) is obtained, it is a lower bound on the optimal objective function value, and we can also define another stopping criteria in terms of the difference between the upper bound and the lower bound.

The decomposition approach algorithm for the dynamic scheduling problem is represented by the flow chart shown in Figure 4-7.

Some Comments on the Decomposition Approach

1. Note that in each iteration, if infeasibilities are identified, we make copies for only a subset of the arcs of the network. The size of the Flow Master Problem in each iteration, is thus contained, thus contributing to tractability.

2. In the conventional approach described in Section 4.7.1, a solution is obtained only when the entire model is solved to completion, which can be an issue due to large computation times in the case of large instances. In contrast, the decomposition approach will have a solution available at the end of each iteration, which can provide potentially useful information to the user.

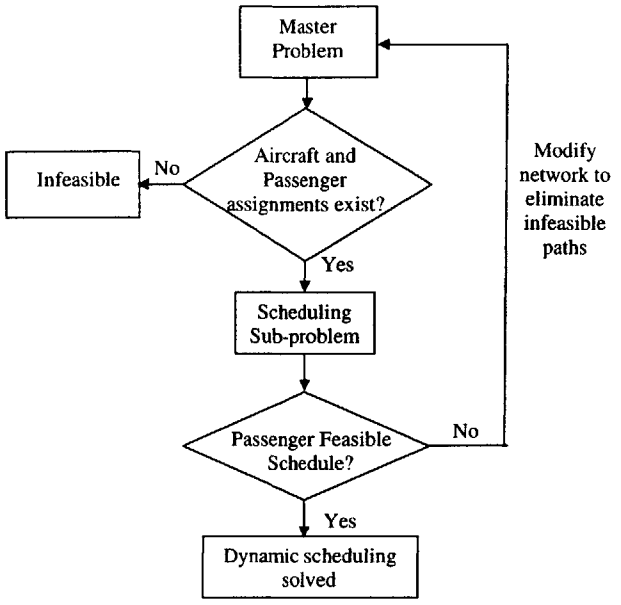

Figure 4-7: Decomposition Solution Approach to Dynamic Airline Scheduling

Computational Experience

In this section, we demonstrate how the issue of scalability is addressed by our decomposition approach. We solve the problem of dynamic scheduling using data from a major US airline. The airline operates a hub-and-spoke network with about 1000 flights that serve approximately 100 cities on a daily basis. Partial passenger data for this network is available and is used as input.

The problem of dynamic scheduling is solved using the re-optimization model of Jiang [21] as well as our decomposition approach, for comparison purposes. We use ILOG CPLEX 9.0 interfaced with Visual Studio C++. Computational experiments are conducted on a computer equipped with one Intel Pentium 4 2.8 GHz processor and 1 GB RAM.

Results and Discussion

For this instance, the decomposition approach required only one iteration, that is, the Flow Master Problem as well as the Scheduling Sub-Problem had to be solved

exactly once, because the maximum profit solution did not require any re-timing for any of the flights, and the solution from the Flow Master Problem had a feasible schedule, as ascertained by the Scheduling Sub-Problem.

These observations were in agreement with the solution obtained from the conventional approach, which was solved by assuming that re-timings up to 15 min are acceptable for each flight. Minute-by-minute copies were made for each flight leg, over an interval of [-15 min, +15 min] from the original flight leg. Note that the routing and fleeting results obtained from the decomposition approach and from the conventional time-space network approach are the same, and both approaches indicated that the original flight schedule need not be re-timed.

We report the problem sizes and computation times from both approaches in Table 4.5. In this table, constraint matrix size is the size of the full formulation in the conventional approach, and the size of the Flow Master Problem in the decomposition approach.

	Time-Space Network	Decomposition
MIP solution time	153.49 sec	0.09 sec
Program Elapsed Time	1059.75 sec	2.56 sec
Constraint Matrix Size	185000x394000	12000x15900

Table 4.5: Comparison between our Decomposition Approach and a Conventional Approach

	Solution Time
Flow Master Problem	0.1 sec
Scheduling Sub-Problem	0.01 sec

Table 4.6: Solution times for modules of the Decomposition approach

Table 4.6 shows the short solution times for both the Flow Master Problem and the Scheduling Sub-Problem, indicating that several iterations of the decomposition approach can be run in the same time that the conventional approach is solved.

Addition of copies to the Flow Master Problem network as the iterations progress can increase the problem size, but the size of the network is still manageable and tractable. In the case of large instances, our decomposition approach can potentially undergo a large number of iterations, each time creating a solution that is 'more' schedule-feasible, whereas conventional approaches do not give indications of progress of the solution algorithm, and therefore in very large instances, the user is unable to get any intermediate information that may be useful.

This computational experiment provides proof-of-concept that our decomposition approach might be effective in addressing large-scale problems that otherwise face issues of intractability.

One such problem that is closely related to dynamic scheduling is robust hub de-peaking, that is, spreading out the departing and arriving flights at a hub to create more uniform demands for runway capacity, gates and airline resources at hubs. De-peaking of hubs is an important planning problem in the airline industry. Jiang [21] presents the robust de-peaking approach as a predecessor of dynamic scheduling, to produce a more robust airline schedule. Jiang reports that the robust de-peaking formulation did not solve in most cases, and restrictive assumptions were imposed to make the models tractable. A natural extension of our work, therefore, is to apply the decomposition approach to the problem of robust hub de-peaking.

4.8 Summary

In this chapter we apply the decomposition approach presented in Section 3.2 to an instance of dynamic airline scheduling. We describe in detail the solution procedures for each of the modules of our decomposition algorithm. Network algorithms to solve the Pre-processing step and the Scheduling Sub-Problem are discussed. We argue that our decomposition approach can capture capacity, cost, demand and supply uncertainties in the Flow Master Problem using the ECCP, and time uncertainty in the Pre-processing and Scheduling Sub-Problem using the CCP. We also discussed

how infeasible solutions generated in the decomposition algorithm can be eliminated, how such constraints can be strengthened, and convergence and complexity issues of our decomposition algorithm. In addition, we show how our decomposition approach is applicable to large-scale problems. Computational results indicate that the output of our decomposition algorithm is 'inherently robust,' providing windows of feasible schedules rather than schedules corresponding to points in time.

Chapter 5

Conclusions and Extensions

Focus in the optimization community has shifted from finding solutions to models using nominal data to finding solutions that are valid under a range of data realizations. Investigations reported in the literature have clearly shown that nominal solutions are not optimal, making the strong case for robust solutions. We consider resource allocation problems, which are prevalent, and develop models and methods to generate robust solutions that are less vulnerable to uncertainty.

In particular, we consider the problem of multi-commodity flows with time-windows under uncertainty, which we refer to as the MCFTW-UU. This problem is at the core of many resource allocation problems, including network design problems.

The issue of introducing robustness into existing models and solutions is of increasing importance in the optimization community in the past decade. The area of developing robust models for resource allocation problems is ripe with challenges and opportunities. Our contribution in this thesis is to provide models and methods capable of producing robust solutions to large-scale multi-commodity flow problems with time-windows, under uncertainty.

- In this thesis, we examined several models aimed at producing robust solutions. We consider the Bertsimas-Sim and the Chance-constrained approaches and describe extensions to these models that allow solution of large-scale problems.

- Focusing on the Extended Chance-Constrained Programming approach (ECCP), we propose a new modeling approach which decomposes the routing and scheduling aspects of the MCFTW problem into a Flow Master Problem and a Scheduling Sub-Problem, thus enhancing tractability. Our decomposition approach can be applied to problems that deal with nominal data as well as problems under uncertainty. Under uncertainty, the Master Problem can be formulated as an ECCP and the Scheduling Sub-Problem can capture uncertainties by altering arc costs and times in a manner similar to that used by chance-constrained approaches.

- Our decomposition algorithm solves the Flow Master Problem using standard integer programming techniques and the Scheduling Sub-Problem using network-based techniques, thus enhancing tractability, even for large-scale problems. Additional side-constraints, such as precedence constraints and time-windows, can be implemented without significant increases in complexity.

- Our decomposition approach makes extensive use of standard network-based algorithms, like shortest-path and simple network search algorithms. We explain the applicability of our approach to large-scale problems for which conventional techniques become intractable.

- Application of our decomposition approach to a problem involving dynamic flight scheduling for a major airline indicates that the Flow Master Problem and Scheduling Sub-Problem can be solved very quickly. This supports our hypothesis that our decomposition approach is scalable, and might be applicable to problems such as robust flight schedule de-peaking; a problem that presents computational challenges using conventional approaches when applied to the problems of large U.S. airlines.

5.1 Future Work

During the course of this work, several directions for future research have been revealed, namely:

- The area of robust optimization is rife with challenges, and several open issues on this subject deserve further investigation. As seen in Chapter 2, there are often questions about what constitutes a robust solution. An important issue is, therefore, to identify the metrics needed to measure solution 'robustness'.

- Another important step is to quantify the 'value of robustness'. This requires more than comparing objective function values for different solutions. Instead, because we must compare realized, not planned, costs of the solutions, it is often necessary to build extensive simulations.

- Problem parameters are often correlated to some degree, causing the realizations of their values to be correlated. Capturing such correlations is an important challenge that, if addressed, should lead to better models and solutions.

- An important goal is to extend the approaches discussed in this thesis to the more complex problem of network design. In the following section, we provide a framework to extend our decomposition approach to network design problems.

- In the scheduling algorithm, we can apply heuristics involving route swaps or local neighborhood searches to generate feasible schedule assignments. This can potentially reduce the number of required iterations of the algorithm to generate a solution.

5.1.1 Network Design

We propose a framework for modeling and solving network design problems with our decomposition approach embedded. In network design problems, decisions involve specifying the network as well as determining optimal flows on the network. We present an algorithmic framework for finding a solution to this problem, and illustrate

our approach for a vehicle routing problem with pickup and delivery time-windows under uncertainty (VRPPDTW-UU), described in Chapter 1. We relax the assumption that the vehicle routes are known so that in the case of the VRPPDTW-UU, routing decisions correspond to both routes of vehicles as well as flows of shipments. Scheduling decisions correspond to the movements of both vehicles and shipments. Costs are incurred due to the routing of both vehicles and shipments. Uncertainties may occur in costs, supplies and demands, capacities, and service times.

In our future work, we plan to embed the decomposition methodology we developed in Chapter 3 into an expanded framework in which vehicle routes are determined, and not known a priori. We propose to use a heuristic method to find vehicle routes and then use our decomposition approach to find shipment routes and schedules. A suitable heuristic must be designed to construct vehicle routes, switch from one 'good' vehicle routing solution to another, and account for variability in vehicle operating costs.

The proposed framework for solving the VRPPDTW-UU is shown in Figure 5-1.

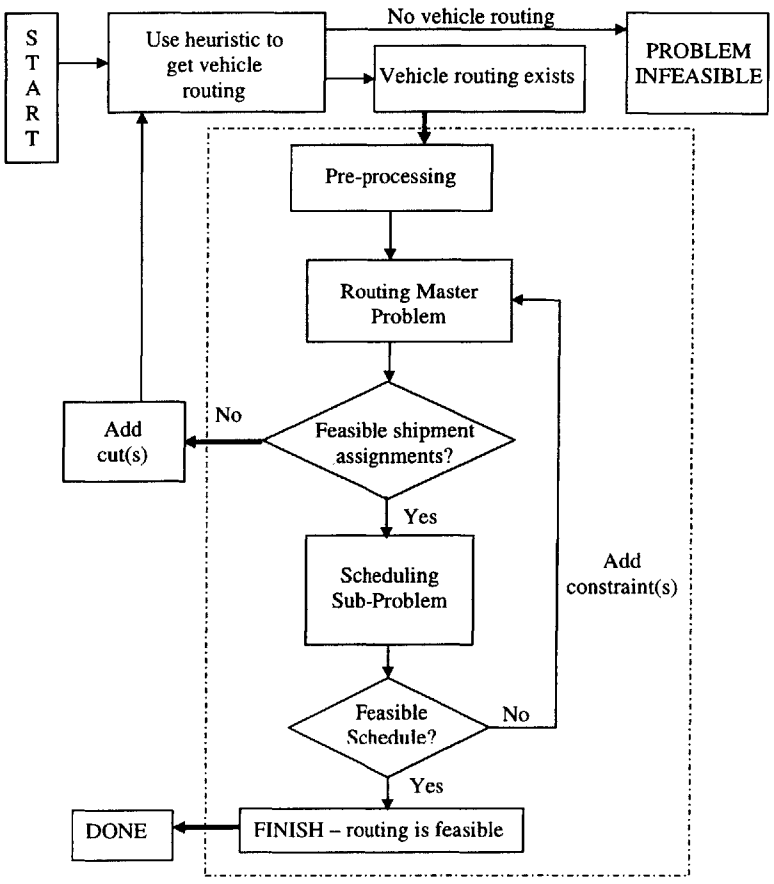

Figure 5-1: Decomposition Approach applied to Network Design

Bibliography

[1] Air traffic association, annual report. 46, 2006.

[2] Ravindra Ahuja, Thomas Magnanti, and James Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, New York, 1993.

[3] C. Barnhart. Airline schedule planning and optimization. stellar.mit.edu, Massachusetts Institute of Technology, 2006. Reading for Course 1.232,15.054/16.71/ESD.217.

[4] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998a.

[5] A. Ben-Israel. *On some problems of mathematical programming.* PhD thesis, Northwestern University, 1962.

[6] A. Ben-Tal and A. Nemirovski. Robust solutions to uncertain programs. *Operations Research Letters*, 25:1–13, 1999.

[7] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

[8] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming Series B*, 98:49–71, 2003.

[9] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[10] D. Bertsimas and A. Thiele. A robust optimization approach to supply chain management. *Massachusetts Institute of Technology*, 2003.

[11] C. Bryant. Robust planning for effects-based operations. Masters thesis, Massachusetts Institute of Technology, 2006.

[12] A. Charnes and W. W. Cooper. Chance constrained programming. *Management Science*, 6(1):73–79, 1959.

[13] A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39, 1963.

[14] A. Charnes, W. W. Cooper, and G. L. Thompson. Critical path analyses via chance constrained and stochastic programming. *Operations Research*, 12(3), 1964.

[15] A. Charnes and M. Kirby. Some special p-models in chance-constrained programming. *Management Science*, 14(3):183195, November 1967.

[16] A. Charnes, M. Kirby, and W. Raike. Chance-constrained generalized networks. *Operations Research*, 14(6):1113–1120, 1966.

[17] A. Charnes and A. Stedry. A chance-constrained model for real-time control in research and development management. *Management Science*, 12(8), 1966.

[18] J. Cordeau, G. Laporte, J. Potvin, and M. Savelsbergh. Transportation on demand. *Handbook of Operations Research*, Section 2, 2006.

[19] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. *Time Constrained Routing and Scheduling, Handbook in Operations Research/ Management Science, Network Routing.* M. Ball, North-Holland, Amsterdam, 1995.

[20] Editors Dongarra et. al. *Netlib Library.* http://www.netlib.org/.

[21] H. Jiang. *Dynamic Airline Scheduling and Robust Airline Schedule De-Peaking.* PhD thesis, Massachusetts Institute of Technology, 2006.

[22] L.S. Kang and J.P. Clarke. Degradable airline scheduling. Working paper, Operations Research Center, Massachusetts Institute of Technology, 2002.

[23] S. Lan, J.P. Clarke, and C. Barnhart. Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28, February 2006.

[24] B. Miller and H. Wagner. Chance constrained programming with joint constraints. *Operations Research*, 13(6):930–945, 1964.

[25] J. M. Mulvey, R. J. Vanderbei, and S. A. Zeinos. Robust optimization of large-scale systems. *Operations Research*, 43:34–56, 1981.

[26] E. Nuutila and E. Soisalon-Soininen. On finding the strongly connected components in a directed graph. *Information Processing Letters*, 49:9–14, 1994.

[27] D. Paraskevopoulos, E. Karakitsos, and B. Rustem. Robust capacity planning under uncertainty. *Management Science*, 37(7):787–800, 1991.

[28] P. Sakamoto. UAV Mission Planning Under Uncertainty. Masters thesis, Massachusetts Institute of Technology, 2006.

[29] M. Savelsbergh. Search in routing problems with time windows. *Annals of Operations Research*, 14:285–305, 1985.

[30] S. Shebalov and D. Klabjan. Robust airline crew pairing: Move-up crews. Working paper, the University of Illinois at Urbana-Champaign, Available from http://netfiles.uiuc.edu/klabjan/www, 2004.

[31] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21, 1973.

[32] R. Tarjan. Depth first search and linear graph algorithms. *Journal of Computing*, 1(2):146–160, 1972.

[33] D. Wood. An algorithm for finding a maximum clique in a graph. *Operations Research Letters*, 21:211–217, 1997.

4224-51