Supply Network Design and Routing for Perishable Commodities

A thesis presented

by

Chris Coey

to

Applied Mathematics

in partial fulfillment of the honors requirements

for the degree of

Bachelor of Arts

Harvard College

Cambridge, Massachusetts

November 20, 2012

Acknowledgements

It is difficult for me to express how much I have gained from the opportunity to learn from and work alongside Professor Özlem Ergun since she came to Harvard last Fall. I am deeply grateful for her guidance and encouragement, and for all the time she has volunteered to mentor me.

I am very thankful to Professor Yiling Chen, whose teaching first sparked my interest in optimization almost three years ago. Professor David Parkes ultimately made everything possible, including the three best classes I took at Harvard, and for that I cannot thank him enough.

Finally, I want to acknowledge my parents: Mum, for staying at home to raise me, and Dad, for supporting me in more ways than one. Thank you both for the trust and the willingness to go along with my crazy ideas, for all the opportunities you've made possible, and for the sacrifices you've made along the way.

Abstract

The United Nations World Food Programme (WFP) distributed over 3.6 million tonnes of food to nearly 100 million people in 75 countries in 2011. After successful trials of a new class of specialized nutrition products (SNPs) for the treatment and prevention of acute malnutrition, the organization is considering how to expand the distribution of these products. The new challenge for the WFP is that these SNPs have limited shelf-lives and need to be refrigerated if stored at warehouses. A two-stage, scenario-based stochastic program is formulated for the fixed-charge network problem of deciding which warehouses to refrigerate and how to route perishable commodities given uncertain demands and network parameters. A Benders decomposition separates the first stage refrigeration problem and the second stage commodity routing problems for each scenario. Following a re-definition of path variables, the second stage problems are solved using a cutting planes procedure and a novel multilabel constrained shortest path pricing algorithm that exploits the problem structure. Computational testing demonstrates that for a realistic problem size of 75 warehouses, 108 time periods, and nearly 200 demand orders in four scenarios, an optimal solution for the refrigeration and commodity routing problem is typically found in around 5 minutes. Future work at the WFP headquarters will involve developing a robust decision tool for operational and tactical level planning on the supply chain for SNPs.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Statement . . . 6
1.3 Optimization Models and Solution Methods . . . 9
1.4 Literature Review and Contributions . . . 11
1.4.1 Humanitarian Logistics and Supply Chains . . . 11
1.4.2 Benders Decomposition and Fixed-Charge Network Problems . . . 12
1.4.3 Resource-Constrained Shortest-Path Algorithms . . . 14

2 A Mathematical Programming Formulation **17**
2.1 Graph Representation of the Distribution Network . . . 17
2.2 Notation for Orders . . . 19
2.3 Paths and Flow Routing Decisions . . . 20
2.4 Scenario-Based Stochastic Modeling . . . 21
2.5 Formulation for the Refrigeration and Commodity Routing Problem . . . 22

3 Tractable Solution Algorithms **26**
3.1 Size of the *RCR* . . . 26
3.2 Benders Decomposition for the *RCR* . . . 27
3.2.1 Cut Generation Algorithm for the Benders Decomposition . . . 31
3.3 Cutting Planes Algorithm for a *2CR-D* . . . 32
3.4 Path Pricing Algorithm for the Second Stage Cutting Planes Procedure . . . 34
3.4.1 Minimizing the Reduced Costs of Paths . . . 34
3.4.2 Satisfying the Shelf-life Expenditure Constraints . . . 36
3.4.3 A Multilabel Algorithm for the Path Pricing Subproblem . . . 36

4 Computational Testing **44**
4.1 Software and Solvers . . . 44
4.2 Network Graph Generation . . . 44
4.3 Stochastic Scenario and Order Generation . . . 45
4.4 Options for Removing Paths in the Second Stage Problem . . . 47
4.5 Benders Subproblems to Solve after a Master Iteration . . . 47
4.6 Testing Scaleability of Algorithms with Different Problem Classes . . . 48

5 Conclusions and Future Work **52**

List of Figures

1 Percentage of Population Undernourished by Country 1
2 World Monthly Food Price Index Since 1990 2
3 WFP Procurement in 2010 3
4 WFP Procurement of SNPs, 2009-2011 4
5 LNS Products Commonly Distributed by the WFP 5
6 United Nations Humanitarian Response Depots, Managed by the WFP . . . 7
7 The Planning Levels of a Supply Chain for Perishable Commodities 9
8 Small Example of a Time-Expanded Network Graph 20
9 Solution Algorithm for the RCR 28
10 An Example Path, Displaying Modified Node and Edge Costs 35
11 An Example Path, Displaying Shelf-Life Expenditures on Edges 37
12 Map of the Suppliers, Warehouses, and Transport Routes used to Construct G 51

List of Algorithms

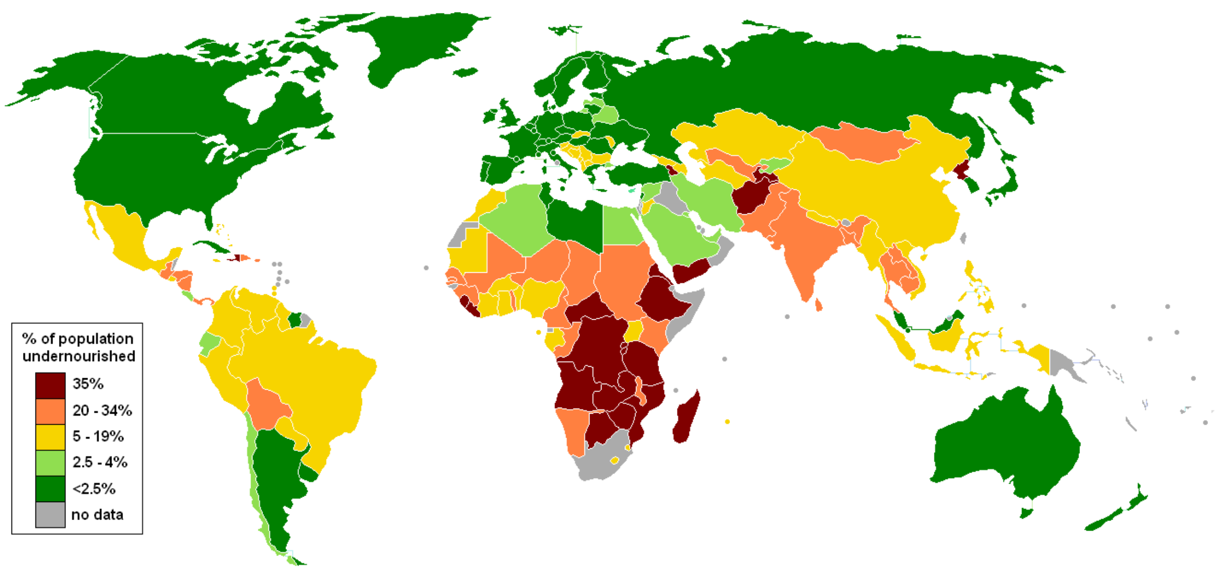
3.1 Benders Decomposition Algorithm for First Stage Refrigeration Problem . . 32
3.2 Cutting Planes Algorithm for Second Stage Commodity Routing Problem . . 33
3.3 Multilabel Constrained Shortest Paths Algorithm 38
3.4 Labels Push Function, part 1 40
3.5 Labels Push Function, part 2 41
3.6 Nondominated Labels Function 42
3.7 Path Construction Function 43

1 Introduction

1.1 Motivation

Hunger is the greatest risk to human health worldwide. According to the Food and Agriculture Organization of the United Nations (FAO), 870 million people worldwide are undernourished and do not have enough food to be healthy and lead an active life. 98 per cent of these people live in developing countries [23]. Under-nourishment causes malnutrition, which the World Health Organization and the United Nations Standing Committee on Nutrition is the largest contributor to disease [38]. Malnutrition at an early age results in redardation of mental and physical development, and more than 147 million pre-schoolers in developing countries are affected by stunted growth [38]. While the number of chronically hungry people declined in the 1980s and the first half of the 1990s, it has risen steadily over the last two decades [23]. Figure 1 illustrates the proportion of the population undernourished in different countries.

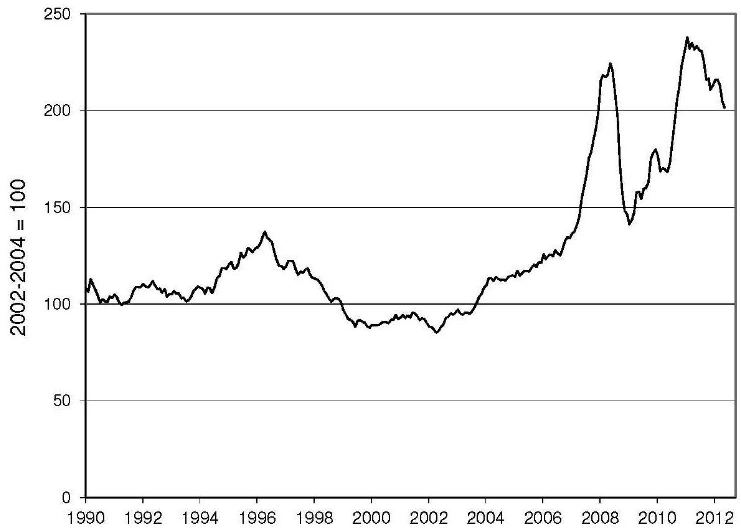
Figure 1: Percentage of Population Undernourished by Country



source: [32]

Organizations such as the FAO, the World Food Programme (WFP), and the United Nations Children's Fund (UNICEF) typically work in unison and with smaller non-governmental organizations to deliver specialized products to treat acute malnutrition. Since need outstrips supply, treatment for children and pregnant or lactating women is prioritized in order to help prevent irreversible developmental disabilities [38]. The global economic downturn and the high and fluctuating food prices since 2008 have meant that international organizations combating hunger have had to increase their efforts and their budgets as an adequate intake of food became too costly for more people. Figure 2 illustrates the trend in the price of a standard food basket providing essential macro- and micro-nutrients. With world food prices expected to reach an all-time high in early 2013 and continue rising thereafter, international organizations such as the WFP are exploring new ways to increase their ability to deliver large quantities of products that are more effective at preventing and treating malnutrition [50].

Figure 2: World Monthly Food Price Index Since 1990



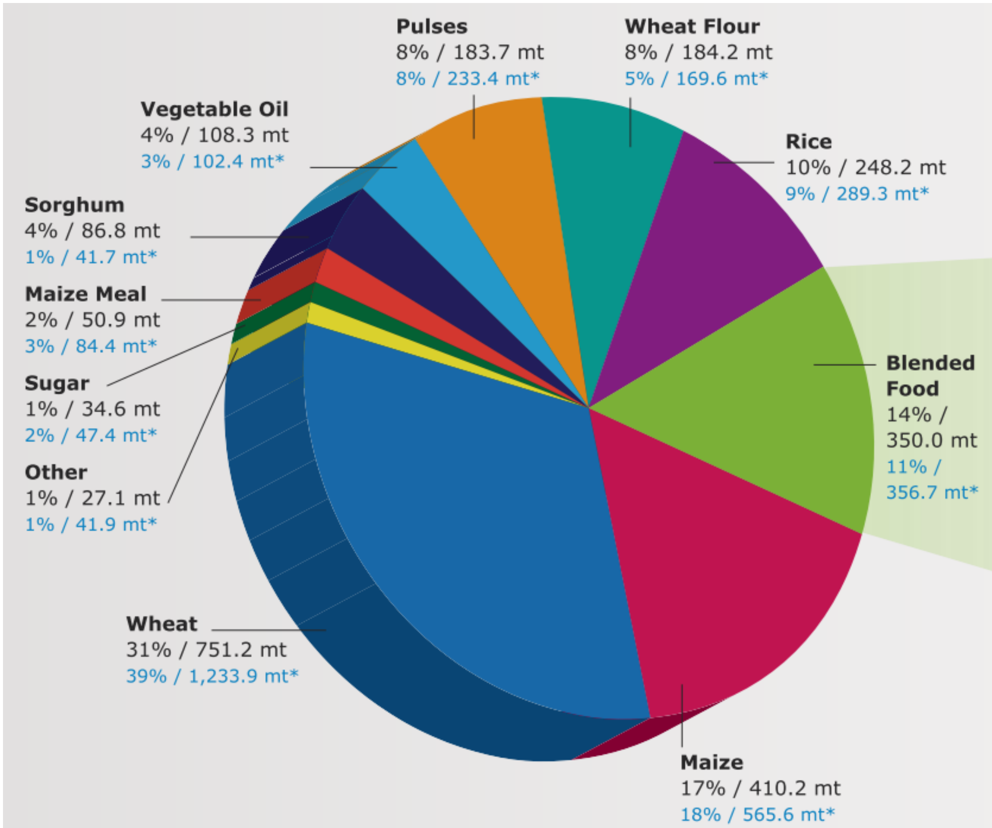
source: [29]

The WFP transports more food annually than any other international organization, through a vast supply chain including 30 ships, 70 planes and helicopters, and 5000 trucks [13]. Shipping is the backbone of the WFP's supply chain, with some 90 percent of its food transported by sea [39]. The WFP partners with around 3000 Non-Governmental Organizations (NGOs), most of which are grassroots organizations with local knowledge and connections, to ensure food is delivered efficiently to the intended beneficiaries. These NGO partners typically receive the goods at final delivery points (FDPs), which are local or regional warehouses at the end of the WFP's supply chain. In 2011, the WFP was able to serve 99.1 million people in 75 countries, providing over 3.6 million tonnes of food. 11 million children received special nutritional support from the WFP.

Since it was first created by the United Nations in 1963, the WFP has primarily focused on providing a basket of food staples including oil, sugar, salt, and one or more cereals or pulses. However in an effort to prevent and treat malnutrition particularly among children, the WFP has focused more over the past few decades on increasing the quantity of 'blended foods' or specialized nutrition products (SNPs) it distributes. Figure 3 illustrates the relative quantities of foods procured by the WFP in 2010. SNPs are manufactured and typically provide a broad set of macro- and micro-nutrients needed to avoid malnutrition. Over the past few decades, the most widely used categories of SNPs were fortified blended foods (e.g. Supercereal), micronutrient powders, and high-energy biscuits and compressed food bars

[42].

Figure 3: WFP Procurement in 2010

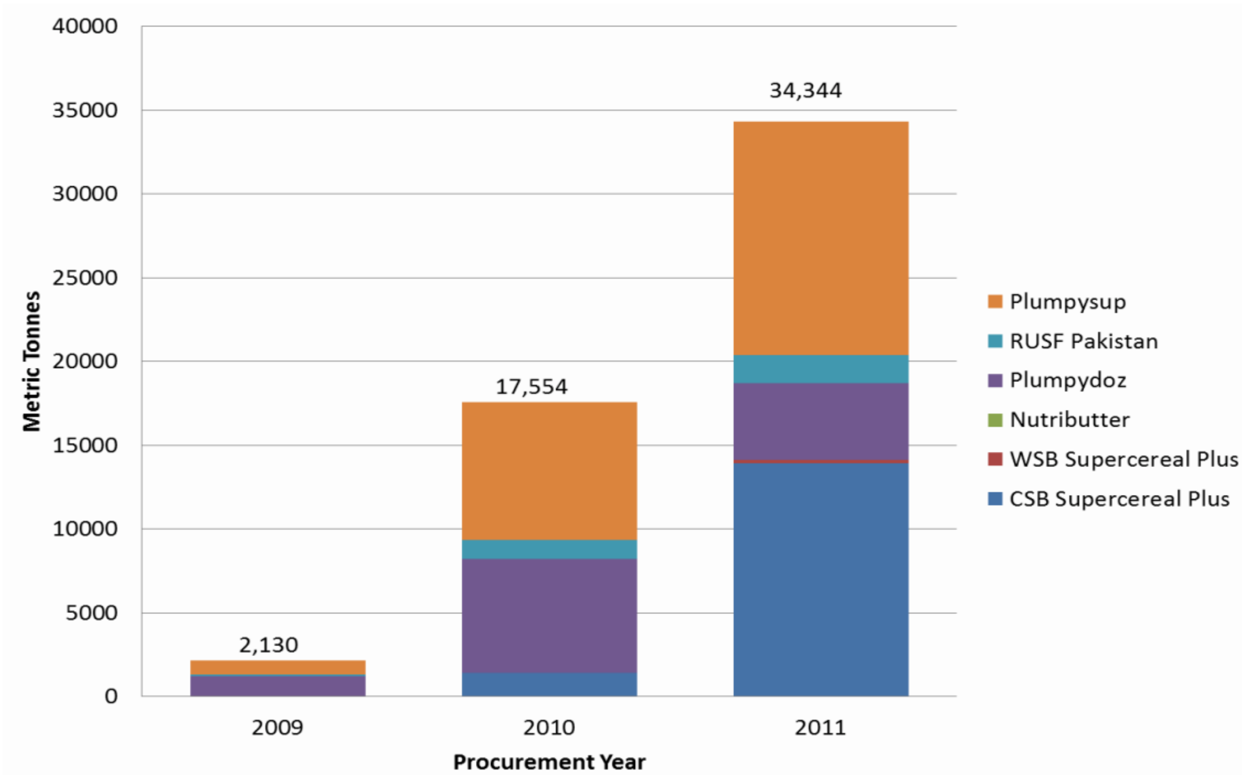


source: [?]

In recent years, a relatively new category of SNP has been shown to hold great promise as an efficient and effective way to prevent and treat malnutrition in children or pregant or lactating mothers. Lipid Based Nutrient Supplements (LNS), also called Ready-to-Use Foods (RUFs), were first conceived of in 1996 when a French company called Nutriset and the Institut de Recherche pour le Développement (IRD) developed the now-famous Plumpy'Nut [35]. Nutriset commercialized the product as an alternative to the therapeutic milks that medical teams such as Médecins Sans Frontières (MSF) administered to malnourished children and mothers. These reconstituted milks required drinking water to be available and had extremely short shelf-lives, so treatment often required hospitalization, severely limiting the number of people that could be treated. On the other hand, Plumpy'Nut is a peanut paste fortified with milk powder, vegetable fats, sugar, and vitamins and other micronutrients and may be consumed at home directly from the packet, without preparation [35].

Between 1997 and 2001, a series of field trials and studies demonstrated that Plumpy'Nut is an effective and efficient way to combat malnutrition. As organizations such as MSF began to use LNS products, Nutriset increased its production [35]. However, it took nearly a decade for larger organizations such as the WFP to begin to acknowledge the advantages of LNSs over the traditional SNPs such as Supercereal, which requires preparation and drinking water, or micronutrient powders, which are supposed to be added to other foods before consumption [42]. In 2005, MSF responded to a food crisis in Niger by treating over 60000 children with Plumpy'Nut and achieving a recovery rate of over 90 percent, demonstrating the effectiveness of LNSs on a large scale [35]. In the years that followed, the WFP began to procure LNSs from Nutriset, and recently the Logistics Development Unit (LDU) of the WFP has been considering how to significantly expand the distribution of these products.

Figure 4: WFP Procurement of SNPs, 2009-2011



source: [43]

Figure 4 shows the recent trend in increasing WFP procurement of new SNPs [43]. Plumpy'Doz and Plumpy'Sup are produced by Nutriset, but unlike Plumpy'Nut, they are formulated for preventing malnutrition rather than treating malnutrition, and Nutributter by Nutriset is essentially Plumpy'Nut. 'RUSF Pakistan' (AchaMum or WawaMum, produced in Pakistan) refers to a new set of LNS products similar to the Plumpy supplements but chickpea based rather than peanut based. The Supercereal Plus products are a variant of the traditional Supercereals and are not lipid-based. More information about LNS and Supercereal products is given in Figure 5.

The major challenge of using SNPs to combat malnutrition is that the products have limited shelf-lives. The shelf-life of a product depends heavily on the temperature and humidity

Figure 5: LNS Products Commonly Distributed by the WFP

Generic Product Term	Lipid-based Nutrient Supplement (LNS) Medium Quantity (20-50g)		Fortified Blended Food (FBF) (100-200g)	
Current WFP Nutrition Products	 Plumpy Doz™ (Peanut-based)	 WawaMum™ (Chickpea-based)	 Supercereal Plus	 Supercereal (mixed with oil and sugar)
Target Group	Children 6-23 months	Children 6-23 months	Children 6-23 months	Pregnant and Lactating Women
Key Ingredients	Vegetable fat, peanut paste, sugar, skim milk powder, whey, sugar, vitamins and minerals	Chickpeas, vegetable oil, milk powder, sugar, vitamins and minerals	Corn or wheat, soya, milk powder, sugar, oil, vitamins and minerals	Corn or wheat, soya, vitamins and minerals
Daily ration	47g	50g	100-200g (200g includes provision for sharing)	100-200g (200g includes provision for sharing)
Nutrient profile	247 kcal, 5.9g protein, 16g fat Essential fatty acids Meets micronutrient requirements	260 kcal, 6.5g protein, 14.5g fat Essential fatty acids Meets micronutrient requirements	420-840 kcal, 16-32g protein, 9-18g fat Essential fatty acids Meets micronutrient requirements	500-1,000 kcal, 17.5-35g protein, 15-30g fat Meets micronutrient requirements
Duration of Intervention[2]	90-180 days	90-180 days	90-180 days	90-180 days
Shelf life	24 months	6 months	12 months	12 months

source: [41]

conditions that it is stored at. Where traditional food staple commodities such as cereals and pulses might be stored for several years in dry conditions, studies analyzing the nutritional content of SNPs over time have shown that if stored at temperatures in excess of $30 - 35^oC$, the products degrade rapidly, some losing important nutrients or separating into oil and solid components in less than six months [44]. A team within the WFP's LDU has been collaborating with academia, partner organizations, and SNP suppliers to develop a comprehensive set of standards for all organizations to store and distribute SNPs. A major focus of this working group is how to prevent the degradation of SNPs before and after delivery, so as to maximize their nutritional benefit and avoid possible spoilage of the goods due to pathogens such as *Cronobacter sakazakii*, which has in several recent cases rendered batches of LNS products too dangerous for consumption [34].

A set of guidelines for temperature and humidity control at WFP and partner organization warehouses is published on the WFP's Food Quality Control website (`foodquality.wfp.org`). For LNSs and other SNPs specifically, the WFP released a document in March 2012, during the Sahel food crisis, describing rapid solutions for modifying warehouses in order to decrease heat and humidity in preparation for the storage of much-needed SNPs [40].

While refrigeration of a warehouse or a compartment within the warehouse provides the ideal dry and cool conditions for optimal preservation of SNPs, electricity is often unavailable or only intermittently available at smaller, local warehouses. Larger warehouses serving whole regions or countries are typically located where roads and other infrastructure is more reliable and electricity is available. Refrigeration is more likely to be feasible at one of these warehouses, which also tend to hold larger quantities of food products and store these products in inventory for long periods of time [40]. The regional level warehouses (extended delivery points, or EDPs) typically distribute to local warehouses (final delivery points, FDPs) only when a need arises at the FDP. If a local NGO served by a particular warehouse places an order for a quantity of some good, units of the good are routed on the distribution network for delivery to the NGO at the warehouse. The economies of scale of refrigerating a larger space for a greater quantity of perishable products such as LNSs in medium-term storage makes refrigerating regional-level warehouses more valuable as the WFP rapidly expands the distribution of such products.

By pre-positioning food products at warehouses, the WFP and partner organizations can respond more rapidly to emergencies that cause hunger. Coordination in the delivery of food resources is particularly challenging and crucial in the aftermath of disasters. The UN Logistics Cluster, led by the World Food Programme, is a group of humanitarian organizations that coordinate and share information to improve the efficiency of logistics responses to humanitarian emergencies [14]. The Logistics Cluster manages the UN Humanitarian Response Depots, five large warehouses displayed in Figure 6 where food and other emergency relief products are stockpiled and can be flown to any location within one or two days. The WFP plans to increase the stockpiles of SNPs at the UNHRDs, increasing the benefit of creating refrigerated sections of these warehouses for medium-term storage of perishable products.

1.2 Problem Statement

In seeking to expand the distribution of SNPs, particularly LNSs such as the peanut or chickpea based fortified pastes, the WFP is faced with the challenge of how to ensure these products do not lose significant nutritional value or succumb to pathogens before consumption by beneficiaries. Since refrigeration better preserves these products, the WFP ought to consider installing costly refrigeration capacity at its country or regional level warehouses, for which the food commodity storage standard is currently adequate ventilation and security only. Following the refrigeration of warehouses, the WFP must route units of goods through the network from suppliers to warehouses to satisfy orders of units requested by WFP field teams or partner organizations. Each order corresponds to a quantity of a particular good that is requested to arrive at a particular time and warehouse location.

The need for SNPs is reflected by the orders that arise on the network. Need has a predictable component, for example the need for nutritional supplementation during seasonal and ongoing famines, and an unpredictable component, due to unexpected events such as natural disasters, political strife, and food price fluctuations. Furthermore, the WFP does not have full information about the activities of its partner organizations. As such, the WFP

Figure 6: United Nations Humanitarian Response Depots, Managed by the WFP

source: [37]

does not know what orders and associated quantities demanded will arise on the network in the future. There may also be uncertaintly about the costs, capacities, and atmospheric conditions on transport routes over time. When making refrigeration decisions, the WFP therefore ought to model orders and edge parameters stochastically.

Given the costs of distribution, limitations on the rate of production by suppliers, and capacities on transport and inventory flows, the WFP is rarely able to satisfy the full quantity of demand for every order. In this case, the organization ought to prioritize orders that it considers more crucial, for example, where the risk of permanent damage to the mental and physical health of babies and young children is high. Furthermore, due to bottlenecks in the supply chain, the WFP will not always be able to deliver goods on-time. For many orders, late deliveries are still valuable compared to no delivery at all, so the WFP must allow for demand to be satisfied late. In the case that some orders cannot be satisfied on-time, the WFP ought to avoid late deliveries for orders for which on-time delivery is very important, and it should be more flexible with delivery times for orders for which lateness is more acceptable.

An optimization model is formulated for the tactical and operational level planning problems that the WFP faces in expanding the distribution of SNPs. The decisions, objectives and constraints for the WFP's optimization problem are outlined below.

The decisions of the WFP are:

1. binary refrigeration decisions on warehouses: which warehouses to install costly refrigeration capacity at before flow routing takes place

2. flow decisions on paths: for each order, how much flow to route on paths that satisfy demand for the order (on-time or late)
3. the quantity of unmet demand for each order

The goals of the WFP are to:

1. minimize the combined cost of refrigerating warehouses, purchasing units, transporting units, and storing units at warehouses
2. minimize the lateness of delivery, or the time between when an order is requested to arrive and the time when it actually arrives
3. minimize unmet demand and where demand outstrips distribution capacity, prioritize meeting demand for the most important orders

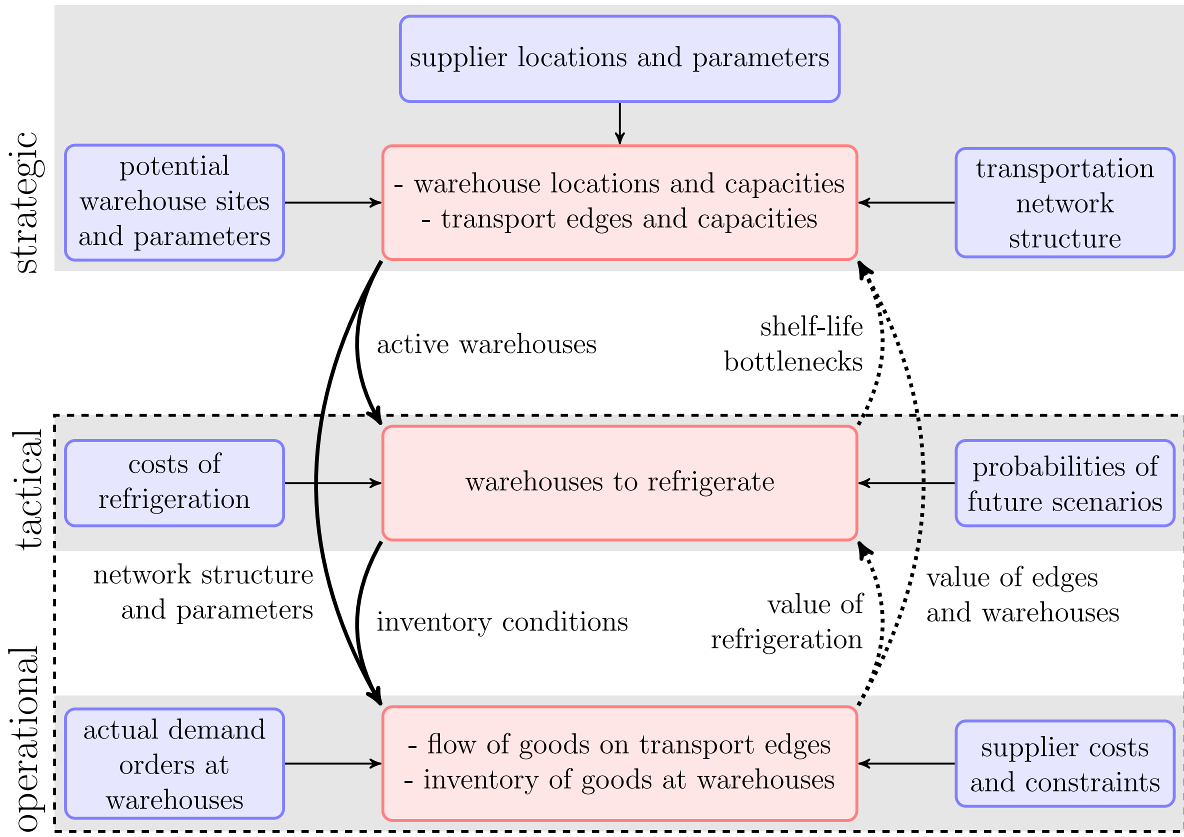
The WFP is constrained by:

1. shelf-lives of the different goods: when a unit is delivered to satisfy an order, it must have a sufficient amount of remaining shelf-life
2. flow capacity of transport routes per unit time: the finite number and capacity of ships, vehicles, and planes limits the quantity that can be transported on each route over a period of time
3. inventory capacity of warehouses: warehouses have limited space for storing goods
4. production capacity of suppliers: suppliers of these manufactured food products are subject to production constraints (except in the long run)

Note that the problem cannot be effectively decomposed into smaller regional subproblems. The WFP operates in 75 countries, but globally there are only several producers of SNPs, and by far the largest of these producers (Nutriset, located in France) holds and enforces a patent on its products. Since global supply is quite limited over the short and medium term time horizons, the WFP's planning problem ought to be solved at the global scale.

The problem facing the WFP can be broken down into subproblems according to the time horizons of the different decisions. Figure 7 illustrates the operational (short-term), tactical (medium-term) and strategic (long-term) planning levels that the WFP faces in expanding its distribution of SNPs. The refrigeration and commodity routing problem described above emcompasses only the operational and tactical levels within the dashed line in Figure 7. The strategic level decisions are simply an example of the network design decisions that might occur at the long-term planning level, such as what locations to open up new warehouses at, what transport routes to construct or secure, and how many ships or vehicles are needed. At the tactical level, information about likely future demand and the costs of refrigerating warehouses, along with information garnered during the process of making routing decisions, is used to decide which of the active warehouses should be refrigerated. At the operational level, routing decisions are made on the fixed network given the fixed refrigeration decisions affecting goods kept in inventory, the actual demand, and the costs and constraints imposed by suppliers of goods.

Figure 7: The Planning Levels of a Supply Chain for Perishable Commodities



1.3 Optimization Models and Solution Methods

The WFP's optimization problem, defined by the decisions, objectives, and constraints listed above, is referred to as the refrigeration and commodity routing (*RCR*) problem. It is formulated as a two-stage stochastic program in which refrigeration decisions are made in the first stage (the *1R* problem) and flow routing and unmet demand decisions are made in the second stage (the *2CR* problem). In the first stage, the orders are unknown, but a distribution over possible future scenarios is available. In the second stage, after refrigeration decisions are fixed, information is revealed and the true scenario becomes known, so a deterministic problem can be solved over the second stage time-horizon. One *2CR* model instance is set up for each scenario.

The *2CR* model is formulated as a path-based multicommodity flow problem with side constraints on a time-expanded network of suppliers, warehouses, transport edges and inventory edges. If a warehouse is selected for costly refrigeration installation, in the second stage units flowing on an inventory edge corresponding to the warehouse will use up a smaller fraction of their allowed shelf-life than if the warehouse was not refrigerated. An optimal solution to a *RCR* model cannot be found by solving the *1R* and *2CR* problems sequentially, since routing information is needed for the refrigeration decisions and vice versa. Since the *RCR* model is extremely large for a distribution network approaching the size of the WFP's, both decomposition and large-scale optimization methods must be implemented.

A path in the second stage problem is re-defined as an ordered list of pairs, where the first element of each pair is an edge in the network and the second is an indicator parameter for whether the edge is an inventory edge associated with a warehouse that the path assumes to be refrigerated. Given a current solution vector of refrigeration decisions, any paths that require a warehouse to be refrigerated are impossible paths if the warehouse is not refrigerated in the solution. This approach allows the shelf-life constraints on paths to be made implicit in the definition of paths, since the shelf-life expended on a path no longer depends on the refrigeration decisions. Without this variable re-definition, the second stage problem cannot be formulated as a continuous linear program and therefore cannot be decomposed effectively for exact large-scale optimization algorithms.

I use Benders decomposition to separate the first and second stage problems in *RCR*. The Benders master problem corresponds to *1R* and one Benders subproblem is created for the *2CR* instance for each possible scenario. A solution to the Benders master problem at some iteration of the decomposition algorithm provides a fixed vector of binary refrigeration decisions to the Benders subproblems. Taking the refrigeration decisions as a vector of parameters, one or more of these subproblems is then solved, and dual information from the optimal solution is passed back to the master problem in the form of a Benders cutting plane. This iterative process eventually reaches an optimal solution for *RCR*. The flow of information between the first and second stage models is illustrated in Figure 7. In one Benders iteration, the tactical level problem *1R* passes refrigeration decisions to an operational level routing problem *2CR*, which takes these decisions as fixed and passes back information on the value of refrigerating each warehouse.

Even after the Benders decomposition, each instance of the *2CR* is too large to solve for a graph approaching the size of the WFP's time-expanded supply chain network. This is because the number of paths is exponential in the size of the network, and increases particularly rapidly following the re-definition of paths. A cutting planes algorithm is used to solve the dual problems of the second stage problems. Violating cuts, corresponding to paths, are generated by a constrained shortest path algorithm that uses dual information from the solution to the second stage problem instance to generate negative reduced cost paths that do not exceed the maximum allowed shelf-life expenditures.

The constrained shortest path algorithm is a highly customized multilabel shortest path algorithm that only stores labels that are nondominated with respect to reduced cost and remaining shelf-life. No labels with nonnegative reduced costs or negative remaining shelf-lives are ever stored. The algorithm adds up to two labels when it 'pushes' label information over an inventory edge, one that assumes the warehouse is refrigerated and the other that assumes it is nonrefrigerated. Although the algorithm is designed to thoroughly exploit the special structure of the directed acyclic graph, the size of this graph makes the multilabel algorithm very computationally expensive, so bounding ideas are also used to discard labels corresponding to sub-paths that cannot be extended into feasible paths. Since the algorithm will always generate at least one lowest reduced cost feasible path, the cutting planes procedure will eventually return an optimal solution for an instance of the *2CR* model.

1.4 Literature Review and Contributions

1.4.1 Humanitarian Logistics and Supply Chains

Thomas and Mizushima [47] defines the field of humanitarian logistics as "the process of planning, implementing and controlling the efficient, cost-effective flow and storage of goods and materials, as well as related information, from the point of origin to the point of consumption for the purpose of meeting the end beneficiary's requirements". This definition was adopted by the WFP and Médecins Sans Frontières. The focus of humanitarian logistics on the needs of the end beneficiary constrasts with the goals of commercial logistics to have "the right item in the right quantity at the right time at the right place for the right price in the right condition to the right customer" [13]. Although the WFP is a donor funded organization without the commercial drive for profit, funding is scarce and the costs of purchasing, transporting and storing LNSs and refrigerating warehouses are far from trivial for the organization. The objective of the *RCR* model is therefore a cost minimization objective where the humanitarian concerns of minimizing unmet demand and minimizing the lateness of deliveries enter as penalization terms.

The tutorial by Celik et al. [13] offers a thorough literature review of recent operations research applications in the logistics of humanitarian disasters. The authors distinguish the relevant literature according to the four phases of disasters: mitigation, preparedness, response, and recovery. The WFP's *RCR* problem falls primarily under the response and recovery phases during which there is a need for SNPs for the prevention and treatment of acute malnutrition. However, the WFP does not exclusively distribute in response to disasters; most of its operations are focused on combating a chronic lack of adequate nutrition. But because of the very rapid onset of urgent need, disasters create significant challenges for the WFP and its partners.

The tutorial also discusses several papers addressing the distribution of aid during the response stage following a disaster. A model for commodity and vehicle routing and the location of temporary facilities in Afshar and Haghani [1] minimizes the unmet demand for different types of supplies needed during the response phase of disasters. Salas et al. [45] describes a stochastic program that minimizes cost and food shortages at a shelter by optimizing the order quantities over time. These models and others that appear in Berkoune et al. [10], Angelis et al. [4], Liberatore et al. [31], and Hentenryck et al. [27] have some methodological similarities to the WFP's *RCR* problem, such as stochastic network parameters and demand, and objectives that weight both cost and demand satisfaction. However, the recent humanitarian logistics literature does not consider in depth the problem of routing perishable goods and making refrigeration decisions on the network.

In an unpublished design project by Alvarenga et al. [3], a shipment scheduling tool was developed in order to improve the WFP's food product routing decisions in the Horn of Africa. The tool consists of a port simulation component and an inland transport corridor optimization component. Donations of food commodities arrive at the port on ships that are queued for unloading. After goods have been unloaded at a port, the cargo is loaded onto vehicles or stored at a port warehouse for later transportation. Based on the port simulations, the tool selects the best port of entry for each shipment. After these decisions are made, an optimization model is used to select land transportation corridors for the shipments on a time-expanded network of warehouses and routes. This capacitated minimum cost flow problem on the time-expanded network is similar to the second stage routing problem of the WFP's *RCR* problem, except that no shelf-life constraints are imposed and thus a simpler edge formulation is used for the multicommodity flow problem. The tool is particularly useful for operational level planning within single regions, where the goods to be routed are the traditional food aid commodities such as cereals and pulses. Unlike the *RCR* problem, it does not make tactical or strategic level decisions that affect the network, and it does not take account of the limited global supply from a small number of producers. It is therefore less suitable for routing the SNPs that are the focus of the WFP's *RCR* problem.

The *RCR* problem and its application to supply chain planning for the World Food Programme is a novel contribution to the field of humanitarian logistics. The development of decision support tools based on this paper will begin in early 2013, and it is hoped that this work will add to the rapidly-growing literature offering operations research solutions to humanitarian issues.

1.4.2 Benders Decomposition and Fixed-Charge Network Problems

The formulation of the *RCR* model as a two-stage stochastic model that is decomposed in a Benders framework is standard. Benders [9] first described the decomposition algorithm, which is based on the ideas of partition and delayed constraint generation. The technique was used in the seminal paper Geoffrion and Graves [26] on multicommodity distribution network design, and in extensions to this work to logistics network design in Cordeau et al. [17]. Tsitsiklis [49] demonstrates how Benders decomposition is can be used to solve two-stage stochastic linear programs. In this broad class of problems, first stage decisions are fixed in the second stage and the optimality of the second stage decisions depends on the first stage decisions.

Benders decomposition can be useful even for deterministic problems if there exists a set of hard linking constraints and a natural separation of variables. Agarwal and Ergun [2] uses Benders decomposition to solve the simultaneous ship-scheduling and cargo-routing (*SSSCR*) problem on a time-expanded network for liner shipping. This is a deterministic model and the Benders algorithm is used simply to separate the tactical and operational level problems in order to arrive at a tractable solution procedure. At the tactical level, corresponding to the first stage Benders master problem, a liner company or alliance selects a set of shipping cycles on which to schedule its ships. At the operational level, corresponding to the single second stage Benders subproblem, the decision-maker routes cargo on the ships to satisfy a set of fixed, known demands at ports.

Whereas the computational difficulty of the decomposed *RCR* problem lies in the second stage *2CR* models, the *SSSCR* has a difficult first stage model. This ship-scheduling model has a binary variable for each cycle in a large time-expanded network of ports, so a column generation approach is used for the integer-relaxed model to generate good cycles between each solve. The relaxation of integrality means that the algorithms proposed for the *SSSCR* do not guarantee an optimal solution. For the *RCR*, however, an optimal solution is guaranteed because the first stage *1R* model has a relatively small number of binary variables and can be solved to optimality quickly without the need for column generation and integer-relaxation.

An example of a stochastic two-stage problem solved with (generalized) Benders decomposition appears in Tsamasphyrou et al. [48]. This paper considers the power systems transmission design problem, in which the goal is to establish a minimum cost set of transmission lines to allow a single commodity, electricity, to flow from generation plants to consumer centers. On a network of 24 nodes and 40 lines, the authors generate up to 2000 scenarios with varying power demands and availabilities and use Benders decomposition to generate cuts. The approach is interesting because the authors consider adding different numbers of Benders cuts between each Benders master problem iteration. They find that the best approach is to divide the scenarios into a number of sets and generate one cut for each set of scenarios for each iteration of the Benders algorithm. This reduces the time taken for the overall algorithm by balancing the computation time needed for the Benders master problem and the many Benders subproblems. A similar analysis is performed with the *RCR* Benders algorithm in the computational testing section of this paper.

The *SSSCR* problem, the power transmission design problem, and the *RCR* problem are all network design problems. General network design problems and applications are surveyed in Magnanti and Wong [33], Gendron et al. [25], Crainic [19], and Costa [18]. More specifically, the *RCR* problem is an example of the capacitated fixed-charge network design problem. Costa [18] characterises this class of network problems as having a set of capacitated edges that may only be traversed if a fixed cost is paid. Fixed-charge network problems are known to be NP-hard [18], so the *RCR* problem is NP-hard. In the *RCR* problem, paying a fixed cost to refrigerate a warehouse effectively opens up new inventory edges at the warehouse with smaller shelf-life expenditures, allowing a potentially larger set of feasible routing decisions to be made on the network. Fixed-charge network design problems have a natural separation of variables that make the Benders decomposition approach suitable. Variables representing the opening of links are solved in the master problem, and flow routing variables are solved for in the subproblem. Thus at each iteration of the algorithm, the master solution gives a tentative network on which the subproblem routes commodities optimally [18].

Typical applications of the capacitated fixed-charge network design problem are in telecommunications networks [25] and electric power transmission networks [11]. Applications of the general fixed-charge network design problem (which may be uncapacitated) often arise in the airline and trucking industries. Armacost et al. [5] describes a model for express package delivery companies such as UPS to establish routes and assign aircraft to the routes at the same time as making routing decisions for packages. This paper is quite unique in that it uses the idea of composite variables to remove the flow decisions and cast the formulation purely in terms of the network design decisions. This result is that the LP relaxations of the composite formulation give stronger lower bounds, speeding up the solution procedure for the MIP. This approach is somewhat similar to the re-definition of path flow variables used in the formulation for the *RCR* problem. The issue of a large number of variables in these models is overcome by solving a restricted problem using a column generation or cutting planes procedure.

1.4.3 Resource-Constrained Shortest-Path Algorithms

For a second stage model corresponding to a particular scenario in the *RCR* problem, the cutting planes procedure that is performed on the dual of the *2CR* uses a pricing subproblem to generate paths corresponding to violated cuts. This pricing subproblem is solved using a resource constrained shortest-path algorithm (*RCSPP*) to ensure that only negative reduced cost paths that do not exceed the shelf-life expenditure constraints are generated. Irnich and Desaulniers [30] describe the general shortest path problem with resource constraints and propose a generic formulation for this class of problems. According to Garey and Johnson [24], even for an acyclic network, positive edge costs, and a single resource constraint - which characterizes pricing subproblem in the *RCR* formulation - the *RCSPP* is known to be NP-hard.

RCSPP and its variants are often encountered in column generation formulations for many classes of problems including the vehicle routing problem and its variants, the fleet assignment model, train scheduling and line planning, and network bandwidth design [22]. Cordeau et al. [16] describes various types of vehicle routing problems and solution methods, and Cordeau et al. [15], Holmberg and Yuan [28], and Sandhu and Klabjan [46] offer recent examples of constrained shortest-path based column generation formulations in transportation and network design problems. In Barnhart et al. [7], the short-haul aircraft rotation problem is solved via column generation of paths in the flight connection network, in which flights are represented as nodes, where paths are constrained so that the total flying time is no longer than that allowed for an aircraft between maintenance stops. Boland et al. [12] develops a label setting algorithm for the elementary resource-constrained shortest path problem, which finds the least cost path with no repeated nodes between two specified nodes subject to the constraint that each resource type consumed by arcs in the path does not exceed its limit. In most of these examples, like for the *RCR* problem, the motivation for solving a variant of the *RCSPP* problem is its usefulness as a subproblem for a column generation or cutting planes procedure.

A common way to solve the *RCSPP* is dynamic programming, which is the approach used in the *RCR* solution framework. The core components of dynamic programming approaches to solving constrained shortest path problems have not changed significantly since the seminal work of Desrochers [20]. Engineer et al. [22] use a dynamic programming algorithm to solve an *RCSPP* as part of a column generation approach for formulations involving very large networks and many resource constraints. The authors give the example of the Dial-a-Flight problem, which arises in the context of an on-demand passenger air transport service operating without a fixed schedule. In this problem, passengers must be assigned and jets must be routed simultaneously, and this creates a set of resource constraints for jet and passenger itineraries that grows rapidly with the number of airports and passengers.

The dynamic programming procedure in Engineer et al. [22] uses a domination scheme for paths, and is therefore similar in some respects to the *RCSPP* algorithm that is used to solve the *RCR* problem. Desrochers and Soumis [21] gives an overview of multilabel shortest path algorithms and the idea of path dominance. Barnhart et al. [6] and Barnhart et al. [8] describe the use of multilabel shortest path algorithms as column generation pricing subproblems for the the crew pairing problem. In this airline crew scheduling context, path dominance is exploited to avoid examining all paths, which correspond to pairings in the crew pairing problem. A similar path dominance approach is used within the multilabel shortest path algorithm to solve the *RCR* problem without enumerating paths in the distribution network.

Irnich and Desaulniers [30] describes dynamic programming algorithms for the *RCSPP*. Starting from the trivial path (*src*), these algorithms systematically build new paths by extending paths one-by-one along all feasible out-edges. The efficiency of an algorithm depends on the ability to detect and discard paths that are not useful for building a Pareto-optimal set of paths. The dominance rules for paths enable the identification of such non-useful paths. Irnich and Desaulniers [30] outlines how paths are encoded by labels (Orlin [36] gives a comprehensive overview of labeling algorithms for dynamic programming approaches, but does not cover multilabel algorithms). Paths sharing a common prefix are represented with a single chain of labels. Each label typically stores a representation of the resource vector.

For the *RCR* problem, the two resources are reduced cost and shelf-life remaining. The dominance rules for the mutlilabel algorithm identify as dominated a path to a particular node that is no better than another path to that node in terms of both resources and strictly worse for at least one resource. Such dominated paths are discarded immediately as non-useful for building the set of nondominated paths. Since the path pricing subproblem for *RCR* requires negative reduced cost paths with nonnegative remaining shelf-lives, two path feasibility conditions are checked in addition to the dominance conditions every time a path is extended in the multilabel algorithm.

What is novel about the path pricing subproblem used to solve the *RCR* problem is the combination of custom techniques to make the multilabel constrained shortest path algorithm more efficient. Tighter bounding ideas that do not significantly increase computation time are used to discard non-useful labels before they become dominated or infeasible. An extreme breadth-first search approach utilizes a property of the time-expanded network to avoid prematurely extending paths that are likely to become dominated is also implemented. These ideas, which rely heavily on the network and model structure, were not seen in the literature on multilabel constrained shortest path subproblems for column generation procedures.

2 A Mathematical Programming Formulation

Section 2.5 presents a mathematical formulation for the RCR problem, along with summaries of the model parameters, sets and variables in Tables 1 to 3. First, the directed graph $G = \langle \mathcal{N}, \mathcal{E} \rangle$ representing the time-expanded distribution network is introduced in Section 2.1, followed by a description of the notation for orders, paths, and stochastic scenarios in Sections 2.2 to 2.4.

2.1 Graph Representation of the Distribution Network

Let $\mathcal{T}$ be the set of discrete time steps in the decision maker's time horizon. A *period* is one of $|\mathcal{T}| - 1$ equal intervals of time spanning the horizon. The granularity of the model with respect to time is arbitrary - a period might correspond to 6 hours, a day, or a week - but it impacts the size of the model. An important consideration for choosing the period length is the range of shelf-lives of the perishable goods. If some goods perish in a matter or days or weeks then the length of a period might be one day or a fraction of a day. Let $\mathcal{G}$ be the set of good types that the decision-maker may distribute, and let $l_g \in \mathbb{R}^+$ be the shelf-life of $g \in \mathcal{G}$. l_g then represents the number of periods that a newly produced unit of g may be stored at ambient conditions before it is judged to be expired. All units purchased from suppliers are assumed to be newly produced.

Let $\mathcal{S}$ be the set of suppliers. A supplier $s \in \mathcal{S}$ can manufacture goods $\mathcal{G}_s \in \mathcal{G}$. For each time step $t \in \mathcal{T}$, s is subject to an aggregate production capacity constraint for all goods and individual production capacity constraints for each $g \in \mathcal{G}_s$. An aggregate production constraint may bind if a supplier's food processing machines may be used to produce different goods and the total output of the machines is limiting. An individual capacity constraint for good g may bind if machines are specialized or if an ingredient only required for g is scarce. These production constraints are imposed by assigning edge flow capacities to two types of supply edges in G. After adding a super-source node *src* and a set of time-expanded suppliers nodes $\mathcal{S} \times \mathcal{T}$ to $\mathcal{N}$, aggregate supply edges $(src, (s,t))\ \forall s \in \mathcal{S}, t \in \mathcal{T}$ are added to $\mathcal{E}$. The flow capacity $m_{(src,(s,t))}$ on edge $(src, (s,t))$ is then equal to the aggregate production capacity for supplier s at time t. To represent the individual good production capacity constraints, a set of good-supplier nodes and a set of individual good production edges are added to G. For each time-expanded supplier node (s,t), nodes $(g,s,t)\ \forall g \in \mathcal{G}_s$ are added to $\mathcal{N}$ and edges $((s,t),(g,s,t))\ \forall g \in \mathcal{G}_s$ are added to $\mathcal{E}$. The flow capacity $m_{((s,t),(g,s,t))}$ on each individual good production edge is then equal to the individual good production capacity for good g by supplier s at t. Also associated with each edge $((s,t),(g,s,t))$ is a flow cost parameter $c_{((s,t),(g,s,t))}$ to account for the full cost of purchasing a unit of good g from supplier s at time t.

After a unit is purchased, the decision-maker routes it from the supplier to a particular warehouse through a network of roads, railways, shipping routes, flight legs, and intermediate warehouses. The set of warehouses $\mathcal{W}$ is time-expanded into the set of nodes $\mathcal{W} \times \mathcal{T} \subset \mathcal{N}$. Since any location where the decision maker may choose to store goods may be modeled as a warehouse, even if the decision-maker does not have the option to refrigerate the facility (for example, a port or partner organization's warehouse), no further nodes are added to $\mathcal{N}$. For a transport route from supplier $s \in \mathcal{S}$ to warehouse $w \in \mathcal{W}$, if $t^{trav} \in \mathbb{Z}^+$ is the discrete number of periods needed for a vehicle to transport units on the route, the set of edges $\{((g, s, t), (w, t')) : g \in \mathcal{G}_s, t \in \mathcal{T}, t' \in \mathcal{T}, t' - t = t^{trav}\}$ is added to $\mathcal{E}$. A unit traversing the transport edge $((g, s, t), (w, t'))$ therefore leaves supplier s at time t and arrives at warehouse w at time t', where the number of periods taken to traverse the edge $t' - t$ is equal to the time needed to traverse the corresponding route from s to w in the distribution network. Similarly, for a route from a warehouse $w \in \mathcal{W}$ to a warehouse $w' \in \mathcal{W} \backslash w$ that takes $t^{trav} \in \mathbb{Z}^+$ periods of transport, the transport edges $\{((w, t), (w', t')) : t \in \mathcal{T}, t' \in \mathcal{T}, t' - t = t^{trav}\}$ are added to $\mathcal{E}$. Note that in order to simplify the process of constructing G, it was assumed that the discrete traversal time for each route remains constant over time. This assumption should be relaxed for routes that have changing traversal times, for example a road that floods seasonally or becomes congested on certain days of the week. For a flight or shipping leg, a transport edge should not be created from a node at time step t if no plane or ship is scheduled to leave the corresponding location at time t.

Since transporting units is costly, each transport edge e is associated with a flow cost parameter $c_e \in \mathbb{R}^+$. Let $m_e \in \mathbb{R}^+$ be the flow capacity parameter equal to the maximum quantity of goods that can be routed during a single period on the corresponding transport route, assumed to be finite. Let $l_e \in \mathbb{R}^+$ be the shelf-life expended on the edge, equal to amount of shelf-life lost during transport on the corresponding route. In general, l_e is not equal to the discrete traversal time of the route because the rate of shelf-life loss depends on the temperature and humidity conditions experienced by units during transport. For example, if edge e corresponds to transport by refrigerated truck on a particular road, l_e will be relatively small compared to $l_{e'}$ for an edge e' corresponding to transport by nonrefrigerated truck on the same road. As implied by this example, in cases where both refrigerated and nonrefrigerated transport options exist for some transport routes, G is a multigraph. To simplify the construction of G, the parameters c_e, m_e, and l_e are assumed to be constant for all edges e corresponding to a particular transport route and mode. This assumption is easily relaxed in the case of periodic or more generally time-dependent transport costs, capacities, and shelf-life expenditures.

The inventory edges at warehouses are the only remaining edge type in $\mathcal{E}$. Inventory edges are crucial in the *RCR* model because of the effect of warehouse refrigeration decisions on the storage conditions for inventoried units. The set $\{((w, t), (w, t + 1)) : w \in \mathcal{W}, t \in \{0, 1, ..., |\mathcal{T}| - 1\}\}$ consists of all edges from a time-expanded warehouse node to another time-expanded warehouse node corresponding to the same warehouse but one time step later. Like transport edges, an inventory edge e has a flow cost parameter c_e, equal to the per-unit, per-period storage cost at the corresponding warehouse, and a flow capacity m_e, equal to the storage capacity of the warehouse. Inventory edge e has two shelf-life expenditure parameters, however. Let l_e represent the shelf-life expended on e under normal, nonrefrigerated conditions, and let l_e^r be the periods of shelf-life that is saved if warehouse

$w(e)$ corresponding to inventory edge e is refrigerated. Therefore $0 \leq l_e^r \leq l_e \; \forall w \in \mathcal{W}, e \in \mathcal{E}(w)$, where $\mathcal{E}(w) = \{((w,t),(w,t+1)) : t \in \{0,1,...,|\mathcal{T}|-1\}\}$ is the set of all inventory edges at warehouse w.

An edge's flow cost, capacity, and life-expenditure parameters are assumed to be equal for all goods. This may seem like an oversimplification of the problem because different goods may be shipped in packages of unequal weights and sizes. However, since the model is continuous, a unit of a good may be defined in terms of kilograms or liters so that transport and storage costs and capacities will not differ widely across goods. For the shelf-life of goods, it is assumed that the perishing rate of goods responds similarly across the spectrum of storage and transport conditions in the network, which is reasonable for a set of quite similar goods.

The directed graph $G = \langle \mathcal{N}, \mathcal{E} \rangle$ that abstractly represents the time-expanded distribution network is illustrated by example in Figure 8 and consists of the following nodes and edges:

$$\mathcal{N} \subset \{src\} \cup (\mathcal{S} \times \mathcal{T}) \cup (\mathcal{G} \times \mathcal{S} \times \mathcal{T}) \cup (\mathcal{W} \times \mathcal{T})$$

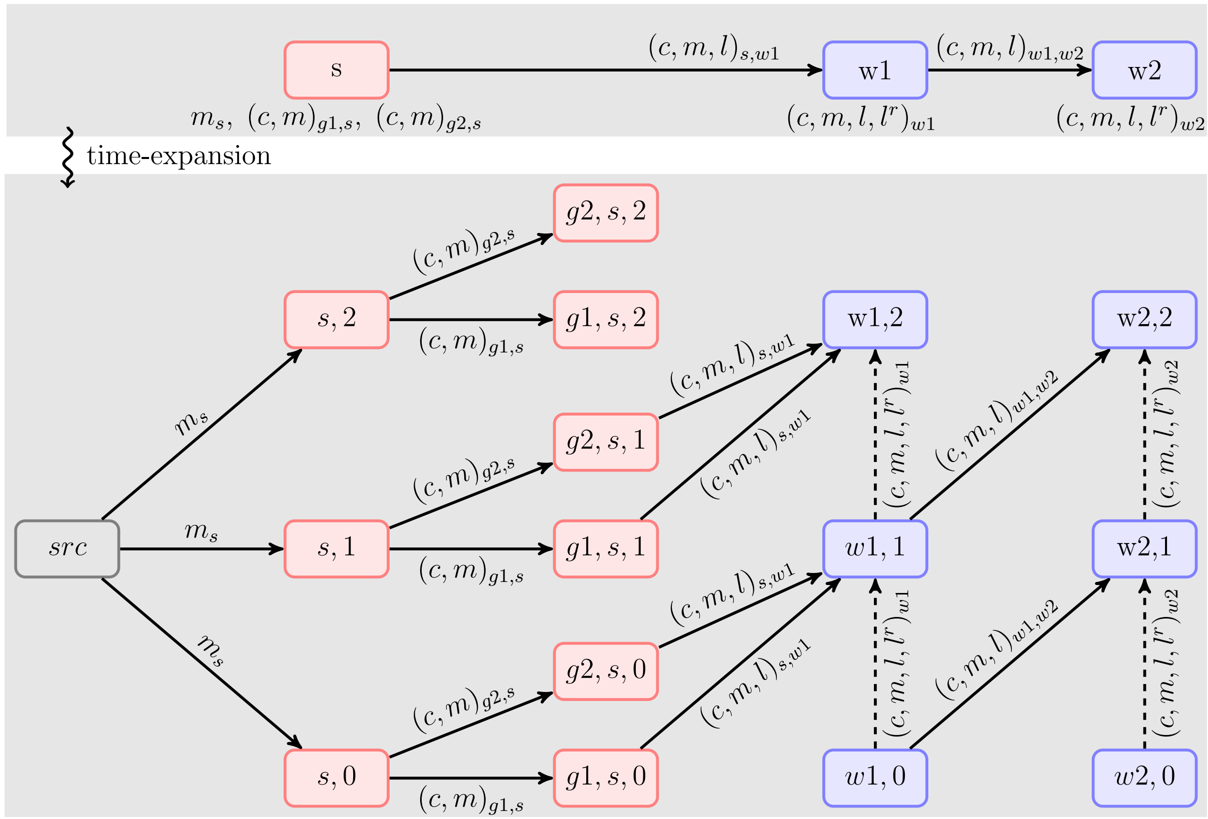
$$\mathcal{E} \subset (\{src\} \times (\mathcal{S} \times \mathcal{T})) \cup ((\mathcal{S} \times \mathcal{T}) \times (\mathcal{G} \times \mathcal{S} \times \mathcal{T})) \cup ((\mathcal{G} \times \mathcal{S} \times \mathcal{T}) \times (\mathcal{W} \times \mathcal{T})) \cup ((\mathcal{W} \times \mathcal{T}) \times (\mathcal{W} \times \mathcal{T}))$$

2.2 Notation for Orders

Demand for units arises at a subset of the time-expanded warehouse nodes. The demand for a particular good g at warehouse node (w,t) is referred to as order (w,t,g), and the set of all orders is $\mathcal{O}$. Each order $o = (w,t,g) \in \mathcal{O}$ is associated with a demand parameter d_o equal to the quantity of the good g demanded at (w,t) and an unmet demand per-unit penalty cost c_o^u that quantifies how important it is to the decision-maker to satisfy units of demand for order o. Order $o = (w,t,g)$ is also associated with a parameter l_o^{max} specifying the maximum periods of shelf-life that can be expended for units being routed from a supplier to the order node (w,t). Since the total shelf-life of good g is l_g, $l_{(g,w,t)}^{max} \leq l_g$. To be useable after an order is fulfilled, units generally need to arrive at warehouse nodes with at least a certain minimum remaining shelf-life, so this inequality will be strict.

The *RCR* model allows for late deliveries for orders but penalizes lateness on a per-unit, per-period basis. This approach, which assumes that the decision-maker's utility for late deliveries declines linearly with the number of periods of lateness, is simple to model. A unit that satisfies a fraction of the demand for order (w,t,g) is delivered late if it is routed through the network to a node in $\{(w,t+1),...,(w,t+t^{late}-1),(w,t+t^{late})\}$, where $t_{(w,t,g)}^{late} \in \mathbb{Z}^+$ is the maximum of allowed late periods for the order. Since G is a capacitated network, it is not always possible to route enough units to node (w,t) to satisfy the full demand $d_{(w,t,g)}$, and even if it is possible to meet the full demand on-time, it may be expensive to do so. In such cases, the allowance for late deliveries may be valuable if the per-unit, per-period penalty cost of lateness for order o, c_o^{late}, is not too high. The parameters should satisfy the

Figure 8: Small Example of a Time-Expanded Network Graph



A small example graph G constructed from a simple case of time periods $0, 1, 2$, two warehouse locations $w1$ and $w2$, and a single supplier s of two goods $g1$ and $g2$. The dashed lines represent inventory edges.

relationship $t_o^{late} \times c_o^{late} \leq c_o^u$, otherwise it would always be better to leave some demand for order $o = (w, t, g)$ unmet than to satisfy it by routing to node $(w, t + t^{late})$.

2.3 Paths and Flow Routing Decisions

A simple, continuous flow routing problem for multiple goods on a network can be solved very quickly with an edge-based multicommodity flow linear program, even for large networks. One aspect of the *RCR* that makes it more difficult to formulate and solve is the constraint on shelf-life expenditure, which ensures that each unit has a sufficient remaining shelf-life at its time of delivery. This constraint cannot be modeled with a simple edge formulation for the multicommodity flow problem because it requires summing over all of the edges that a unit is routed over.

In the path formulation for the multicommodity flow problem, flow routing decisions are made on paths through the network rather than on edges. A path is typically defined as an ordered list of nodes or edges that are traversed from an origin to destination in the network.

Since a path formulation keeps track of the edges that each unit is routed over, a shelf-life expenditure constraint can be incorporated into such a formulation. However, in the *RCR* problem, the shelf-life expended on inventory edges depends on the warehouse refrigeration decisions, which are made simultaneously with the routing decisions. Under a particular vector of refrigeration decisions, a path that routes goods to satisfy an order o may have a total shelf-life expenditure not exceeding the allowed l_o^{max}, but under a different vector of refrigeration decisions, the path may exceed the allowed shelf-life expenditure.

This means the constraints on shelf-life expenditure must link the binary refrigeration variables and the continuous flow routing variables in a complex way to ensure that there is zero flow on paths that exceed the shelf-life expenditure. One possible formulation requires an additional set of binary variables to indicate whether paths violate l^{max} and an additional set of constraints to set flow on each violating path to zero. Another formulation requires an additional set of continuous variables to measure the positive difference between each path's shelf-life expenditure and l^{max} and a nonlinear term that multiplies these variables and the flows on paths. However, neither of these formulations are amenable to decomposition or large-scale optimization techniques, which are necessary because the number of paths and thus the number of flow variables is exponential in the size of G.

To overcome these modeling difficulties, a path is re-defined so that the shelf-life expended on a path becomes a parameter that is invariant to the refrigeration decisions. Now, a path p is an ordered list of pairs $(e, r_{p,e})$, where $e \in \mathcal{E}$ is an edge traversed by the path and $r_{p,e} \in \{0, 1\}$ is an indicator parameter for whether edge e is an inventory edge associated with a warehouse that p assumes to be refrigerated. Any path p that requires a warehouse w to be refrigerated (i.e. $r_{p,w(e)} = 1$ for some edge e traversed by p) is an impossible path if w is not refrigerated. This variable re-definition approach makes the shelf-life expenditure constraints implicit in the formulation by not creating a flow variable for any path p that has a shelf-life expenditure exceeding $l_{o(p)}^{max}$, where $o(p)$ is the order that p routes units to satisfy. The new formulation for the *RCR* problem has a very simple constraint linking warehouse refrigeration decisions and path flow decisions.

2.4 Scenario-Based Stochastic Modeling

The *RCR* is formulated as a two-stage scenario-based stochastic model. Let c_w^r be the fixed, one-off cost of refrigerating warehouse w in the first stage. Let $\mathcal{K}$ be the set of scenarios representing a subset of the possible outcomes for orders and edge parameter values for the second stage. Each scenario will have a different set of orders and therefore also a different set of paths satisfying orders, so these sets are indexed over the scenarios. To simplify notation, an order $o \in \mathcal{O}_k$ exists only in scenario k, even if the tuple $(w, t, g) = o$ matches the tuple for an order in a different scenario. Similarly, a path $p \in \mathcal{P}_k$ exists only in scenario k and is different from a path with the same ordered list of pairs that exists in a different scenario. Since the edge parameters may differ across scenarios, these parameters are explicitly indexed by scenario except where the index is obvious from context.

The decision-maker has a probability distribution over the scenarios, defined by the scenario probabilities $0 \le \alpha_k \le 1$ for $k \in \mathcal{K}$ that satisfy $\sum_{k \in \mathcal{K}} \alpha_k = 1$. Using the scenario probabilities, the decision-maker optimizes an expectated value over the scenarios. The expected value objective ensures that a scenario that is more likely to occur is given proportionally more weight in the decision-making process.

2.5 Formulation for the Refrigeration and Commodity Routing Problem

Table 1: Sets

$\mathcal{W}$	warehouses
$\mathcal{E}$	edges in G
$\mathcal{E}(w)$	inventory edges at warehouse $w \in \mathcal{W}$
$\mathcal{K}$	scenarios in the stochastic model
$\mathcal{O}_k$	orders in scenario $k \in \mathcal{K}$
$\mathcal{P}_k$	paths valid for orders in scenario $k \in \mathcal{K}$
$\mathcal{P}_k(e)$	paths that include edge $e \in \mathcal{E}$ in scenario $k \in \mathcal{K}$
$\mathcal{P}_k(o)$	paths valid for order $o \in \mathcal{O}_k$ in scenario $k \in \mathcal{K}$

Table 2: Parameters

α_k	$\forall k \in \mathcal{K}$	$\ge 0, \le 1$, probability of scenario k
$r_{p,e}$	$\forall p \in \mathcal{P}_k, e \in \mathcal{E}$	1 if e is an inventory edge assumed refrigerated on p, else 0
c^r_w	$\forall w \in \mathcal{W}$	fixed cost of refrigeration for warehouse w
c^f_p	$\forall p \in \mathcal{P}_k$	unit cost of flow on path p used in scenario k
c^u_o	$\forall o \in \mathcal{O}_k$	unit penalty for unmet demand for order o in scenario k
d_o	$\forall o \in \mathcal{O}_k$	units of good demanded in order o in scenario k
$m_{k,e}$	$\forall k \in \mathcal{K}, e \in \mathcal{E}$	flow capacity on edge e in scenario k
$\hat{m}_{k,w}$	$\forall k \in \mathcal{K}, w \in \mathcal{W}$	all-periods flow capacity at warehouse w in scenario k

The flow cost per unit on path p is the sum of the costs of the edges on the path plus the lateness cost for the path. This parameter is used for notational convenience and is easily calculated during the process of constructing paths. The lateness cost for a path satisfying order $o(p) = (w, t, g)$ with destination node $(w, t + \tau), \tau \in \{1, \ldots, t^{late}_{o(p)}\}$ is $\tau \times c^{late}_{o(p)}$, where c^{late}_o is the cost per late period for order o. So for a path p that delivers goods τ periods late for order $o(p)$ in scenario k, the flow cost on p is as follows.

$$c^f_p = \sum_{e \in \mathcal{E}(p)} c_{e,k} + \tau c^{late}_{o(p)} \quad \forall k \in \mathcal{K}, p \in \mathcal{P}_k$$

The parameter $\hat{m}_{k,w}$ for the all-periods inventory flow capacity at warehouse w in scenario k is a big-M type parameter calculated by summing the flow capacities in k of the inventory edges $((w,t),(w,t+1))$ at w.

$$\hat{m}_{k,w} = \sum_{t=0}^{T-1} m_{k,((w,t),(w,t+1))} = \sum_{e \in \mathcal{E}(w)} m_{k,e} \quad \forall k \in \mathcal{K}, w \in \mathcal{W}$$

Table 3: Variables

R_w	$\in \{0,1\}$	$\forall w \in \mathcal{W}$	$= 1$ iff warehouse w is refrigerated
F_p	≥ 0	$\forall p \in \mathcal{P}_k$	flow on path p used in scenario $k \in \mathcal{K}$
U_o	≥ 0	$\forall o \in \mathcal{O}_k$	unmet demand for order o in scenario $k \in \mathcal{K}$

Note that if the decision maker cannot make a refrigeration decision on a particular warehouse, the variable associated with that warehouse is simply fixed to represent whether the storage location is refrigerated or not. For example if a storage location w is owned by a partner organization that does not allow the decision-maker to install refrigeration capacity at the facility, then the decision variable R_w is fixed to 1 if the facility is already refrigerated or 0 if not. Similarly, if the decision-maker is unable to refrigerate a warehouse w due to some constraint about the warehouse then $R_w = 0$ is fixed, and if w is already refrigerated, $R_w = 1$ is fixed.

Model *RCR*: Full Refrigeration and Commodity Routing Problem

$$\begin{aligned}
\min \quad & \sum_{w \in \mathcal{W}} c^r_w R_w + \sum_{k \in \mathcal{K}} \alpha_k \left(\sum_{p \in \mathcal{P}_k} c^f_p F_p + \sum_{o \in \mathcal{O}_k} c^u_o U_o \right) \\
\text{s.t.} \quad & \hat{m}_w R_w - \sum_{e \in \mathcal{E}(w)} \sum_{p \in \mathcal{P}_k(e)} r_{p,e} F_p \geq 0 && \forall k \in \mathcal{K}, w \in \mathcal{W} \\
& \sum_{p \in \mathcal{P}_k(e)} F_p \leq m_e && \forall k \in \mathcal{K}, e \in \mathcal{E} \\
& \sum_{p \in \mathcal{P}_k(o)} F_p + U_o = d_o && \forall k \in \mathcal{K}, o \in \mathcal{O}_k \\
& R_w \in \{0,1\} && \forall w \in \mathcal{W} \\
& F_p \geq 0 && \forall k \in \mathcal{K}, P \in \mathcal{P}_k \\
& U_o \geq 0 && \forall k \in \mathcal{K}, o \in \mathcal{O}_k
\end{aligned}$$

The objective function minimizes the total expected cost. The first component is the sum of the refrigeration costs for warehouses that are selected to be refrigerated. The second component is an expectation weighted by the scenario probabilities $alpha_k, k \in \mathcal{K}$ of the cost of routing the commodities on the logistics network and the penalty cost for demand going unmet.

The second constraint set limits, for each edge $e \in \mathcal{E}$, the total flow on paths including the edge $\sum_{p \in \mathcal{P}_k(e)} F_p$ to the capacity of the edge m_e. The third constraint set specifies, for each order $o \in \mathcal{O}_k, k \in \mathcal{K}$, the relationship between the quantity demanded d_o, the unmet demand U_o, and the demand satisfied due to flows on paths valid for the order $\sum_{p \in \mathcal{P}_k(o)} F_p$. Since the unmet demand and path flows are nonnegative, the quantity of goods satisfying an order can never exceed the demand.

The first constraint set ensures that if a warehouse $w \in \mathcal{W}$ is not refrigerated ($R_w = 0$), the total flow over w's inventory edges $e \in \mathcal{E}(w)$ on paths for which w is hypothetically refrigerated for e is 0. For a particular warehouse w, this means that if w is not refrigerated then there can be no flow on any paths that traverse inventory edges that are assumed to be refrigerated at w. This a big-M type constraint that is tight because it acknowledges that the maximum possible flow on all inventory edges assumed to be refrigerated for a warehouse w is equal to the total all-periods flow capacity summed over the inventory edges $e \in \mathcal{E}(w)$.

Note that it is not necessary to define a constraint in the opposite direction that ensures zero flow on all paths that assume a warehouse w is nonrefrigerated when $R_w = 1$. This is because these paths are feasible for either refrigeration state of w, since refrigerating an additional warehouse while keeping all other refrigeration decisions constant can never increase the shelf-life expenditure of paths.

Note that this formulation for the warehouse refrigeration constraints is aggregated over the inventory edges $e \in \mathcal{E}(w)$ at each warehouse w. A tighter, disaggregated formulation for the refrigeration constraints limits flow on each individual inventory edge assumed to be refrigerated to 0 when the associated warehouse w is not refrigerated, and has the same effect when refrigeration variables are strictly binary (not relaxed). There are $|\mathcal{W}| \times (|\mathcal{T}| - 1)$ disaggregated capacity constraints or $|\mathcal{W}|$ aggregated capacity constraints. The big-M parameter on a disaggregated constraint is the flow capacity on the associated inventory edge $e \in \mathcal{E}$, m_e.

$$m_e R_w - \sum_{p \in \mathcal{P}(e)} F_p \geq 0 \quad \forall w \in \mathcal{W}, e \in \mathcal{E}(w)$$

If solving the full MIP above, using the disaggregated constraint set may yield better solution time and quality because the LP relaxations in the branch-and-bound tree could be significantly tighter. However, the aggregated constraint set is used because following a Benders decomposition of the *RCR* model there will be no benefit from disaggregation, so the smaller constraint set is preferred.

The *RCR* model has a defined structure that is better visualized in matrix-vector format. Table 4 defines matrices representing the coefficients on variables in the constraint matrix of the *RCR* model. The *RCR* model is then rewritten to illustrate the block structure of the constraint matrix.

Table 4: Matrix Representations of Variable Coefficients in Constraints

$\hat{M}_k$	$\lvert\mathcal{W}\rvert \times \lvert\mathcal{W}\rvert$	diagonal square matrix with the entries of $\hat{m}_{k,w}$ $\forall w \in \mathcal{W}$
WP_k	$\lvert\mathcal{W}\rvert \times \lvert\mathcal{P}_k\rvert$	(w, p) is the number of inventory edges at w on p assumed refrigerated
EP_k	$\lvert\mathcal{E}\rvert \times \lvert\mathcal{P}_k\rvert$	(e, p) is 1 iff edge e is traversed by p
OP_k	$\lvert\mathcal{O}_k\rvert \times \lvert\mathcal{P}_k\rvert$	(o, p) is 1 iff p is valid for order o (columns have a single 1)
IO_k	$\lvert\mathcal{O}_k\rvert \times \lvert\mathcal{O}_k\rvert$	the identity matrix with dimension equal to the number of orders

Model *RCR*: Full Refrigeration and Commodity Routing Problem

$$
\begin{array}{llllllll}
\min & \mathbf{c^r R} + \alpha_1(\mathbf{c_1^f F_1} + \mathbf{c_1^u U_1}) + \alpha_2(\mathbf{c_2^f F_2} + \mathbf{c_2^u U_2}) \cdots + \alpha_k(\mathbf{c_k^f F_k} + \mathbf{c_k^u U_k}) & & & & & & \\
\text{s.t.} & \hat{M}_1\mathbf{R} - WP_1\mathbf{F_1} & & & & & & \geq \mathbf{0_1} \\
 & & EP_1\mathbf{F_1} & & & & & \leq \mathbf{m_1} \\
 & & OP_1\mathbf{F_1} + IO_1\mathbf{U_1} & & & & & = \mathbf{d_1} \\
 & \hat{M}_2\mathbf{R} & & - WP_2\mathbf{F_2} & & & & \geq \mathbf{0_2} \\
 & & & EP_2\mathbf{F_2} & & & & \leq \mathbf{m_2} \\
 & & & OP_2\mathbf{F_2} + IO_2\mathbf{U_2} & & & & = \mathbf{d_2} \\
 & \vdots & & & \ddots & & & \vdots \\
 & \hat{M}_k\mathbf{R} & & & & - WP_k\mathbf{F_k} & & \geq \mathbf{0_k} \\
 & & & & & EP_k\mathbf{F_k} & & \leq \mathbf{m_k} \\
 & & & & & OP_k\mathbf{F_k} + IO_k\mathbf{U_k} & & = \mathbf{d_k}
\end{array}
$$

$$
R_w \in \{0,1\} \; \forall w \in \mathcal{W}, \quad F_p \geq 0 \; \forall k \in \mathcal{K}, P \in \mathcal{P}_k, \quad U_o \geq 0 \; \forall k \in \mathcal{K}, o \in \mathcal{O}_k
$$

The number of rows in the constraint matrix of the *RCR* model is $|\mathcal{K}| \times (|\mathcal{W}| + |\mathcal{E}|) + \sum_{k \in \mathcal{K}} |\mathcal{O}_k|$ and the number of columns is $|\mathcal{W}| + \sum_{k \in \mathcal{K}} (|\mathcal{P}_k| + |\mathcal{O}_k|)$. Note that the binary refrigeration variables link all of the scenarios through the refrigeration constraints. In the next section, Benders decomposition is used to overcome the hardness of these linking constraints. Since the number of paths in each scenario is exponential in the size of G, a large-scale optimization algorithm is also implemented.

3 Tractable Solution Algorithms

This section develops the entire solution algorithm for the *RCR* formulation after discussing its size in Section 3.1. Section 3.2 describes the Benders decomposition algorithm for separating the first stage refrigeration problem (*1R*) from the second stage commodity routing problem (*2CR*) for each scenario $k \in \mathcal{K}$. Section 3.3 describes the cutting planes algorithm used to solve a second stage commodity routing problem to optimality using the multilabel constrained path pricing algorithm. Table 5 summarizes the notation for first and second stage model names. Tables 6 to 8 summarize the model sets, parameters, and variables used in this section. Figure 9 provides a high-level representation of the entire solution algorithm for the *RCR* formulation. When contextually obvious, indexing of models, parameters and sets by scenario is dropped for notational convenience.

Table 5: Model Names and Notation

1R	full first stage refrigeration problem
1R-R	first stage refrigeration problem with a restricted set of cuts
2CR	full second stage commodity routing primal problem
2CR(k)	full second stage commodity routing primal problem for scenario $k \in \mathcal{K}$
2CR($\bar{\mathbf{R}}$)	second stage problem given a vector $\bar{\mathbf{R}}$ of first stage refrigeration decisions
2CR-D	full second stage commodity routing dual problem
2CR-DR	second stage commodity routing dual problem with a restricted set of cuts

Table 6: Sets

$\mathcal{W}$	warehouses
$\mathcal{E}$	edges in G
$\mathcal{E}(w)$	inventory edges at warehouse $w \in \mathcal{W}$
$\mathcal{K}$	scenarios in the stochastic model
$\mathcal{O}_k$	orders in scenario $k \in \mathcal{K}$
$\mathcal{P}_k$	paths valid for orders in scenario $k \in \mathcal{K}$
$\mathcal{P}_k(e)$	paths that include edge $e \in \mathcal{E}$ in scenario $k \in \mathcal{K}$
$\mathcal{P}_k(o)$	paths valid for order $o \in \mathcal{O}_k$ in scenario $k \in \mathcal{K}$

3.1 Size of the *RCR*

The full refrigeration and commodity routing problem *RCR* has $|\mathcal{W}|$ binary refrigeration variables, $\sum_{k \in \mathcal{K}} |\mathcal{P}_k|$ continuous path flow variables, and $\sum_{k \in \mathcal{K}} |\mathcal{O}_k|$ continuous unmet demand variables. The number of edges in the multigraph G can be large. Even before paths were re-defined, the number of standard paths in G is exponential in the size of G. Following the re-definition of a path p as an ordered list of pairs $(e, r_{p,e})$, where $e \in \mathcal{E}$ is an edge traversed by the p and $r_{p,e} \in \{0, 1\}$ is an indicator parameter for inventory edge refrigeration, each standard path of length n edges corresponds to almost 2^n re-defined paths in the worst

Table 7: Parameters

α_k	$\forall k \in \mathcal{K}$	$\geq 0, \leq 1$, probability of scenario k
$r_{p,e}$	$\forall p \in \mathcal{P}, e \in \mathcal{E}$	1 if e is an inventory edge assumed refrigerated on p, else 0
c^r_w	$\forall w \in \mathcal{W}$	fixed cost of refrigeration for warehouse w
c^f_p	$\forall p \in \mathcal{P}$	unit cost of flow on path p
c^u_o	$\forall o \in \mathcal{O}$	unit penalty for unmet demand for order o
d_o	$\forall o \in \mathcal{O}$	units of good demanded in order o
$m_{k,e}$	$\forall k \in \mathcal{K}, e \in \mathcal{E}$	flow capacity on edge e in scenario k
$\hat{m}_{k,w}$	$\forall k \in \mathcal{K}, w \in \mathcal{W}$	all-periods flow capacity at warehouse w in scenario k
$\bar{R}$	$\forall w \in \mathcal{W}$	refrigeration decision on w passed to a second stage problem

Table 8: Variables

R_w	$\in \{0, 1\}$	$\forall w \in \mathcal{W}$	$= 1$ iff warehouse w is refrigerated
F_p	≥ 0	$\forall p \in \mathcal{P}$	flow on path p
U_o	≥ 0	$\forall o \in \mathcal{O}$	unmet demand for order o
ρ_w	≤ 0	$\forall w \in \mathcal{W}$	dual price on refrigeration constraint for w
μ_e	≤ 0	$\forall e \in \mathcal{E}$	dual price on edge flow capacity constraint for e
δ_o	≥ 0	$\forall o \in \mathcal{O}$	dual price on order demand constraint for o
Z_k	≥ 0	$\forall k \in \mathcal{K}$	maximum second stage cost for scenario k in *1R-R*
Z'_k	≥ 0	$\forall k \in \mathcal{K}$	optimal objective value of *2CR-D(k)*($\bar{\mathbf{R}}$)

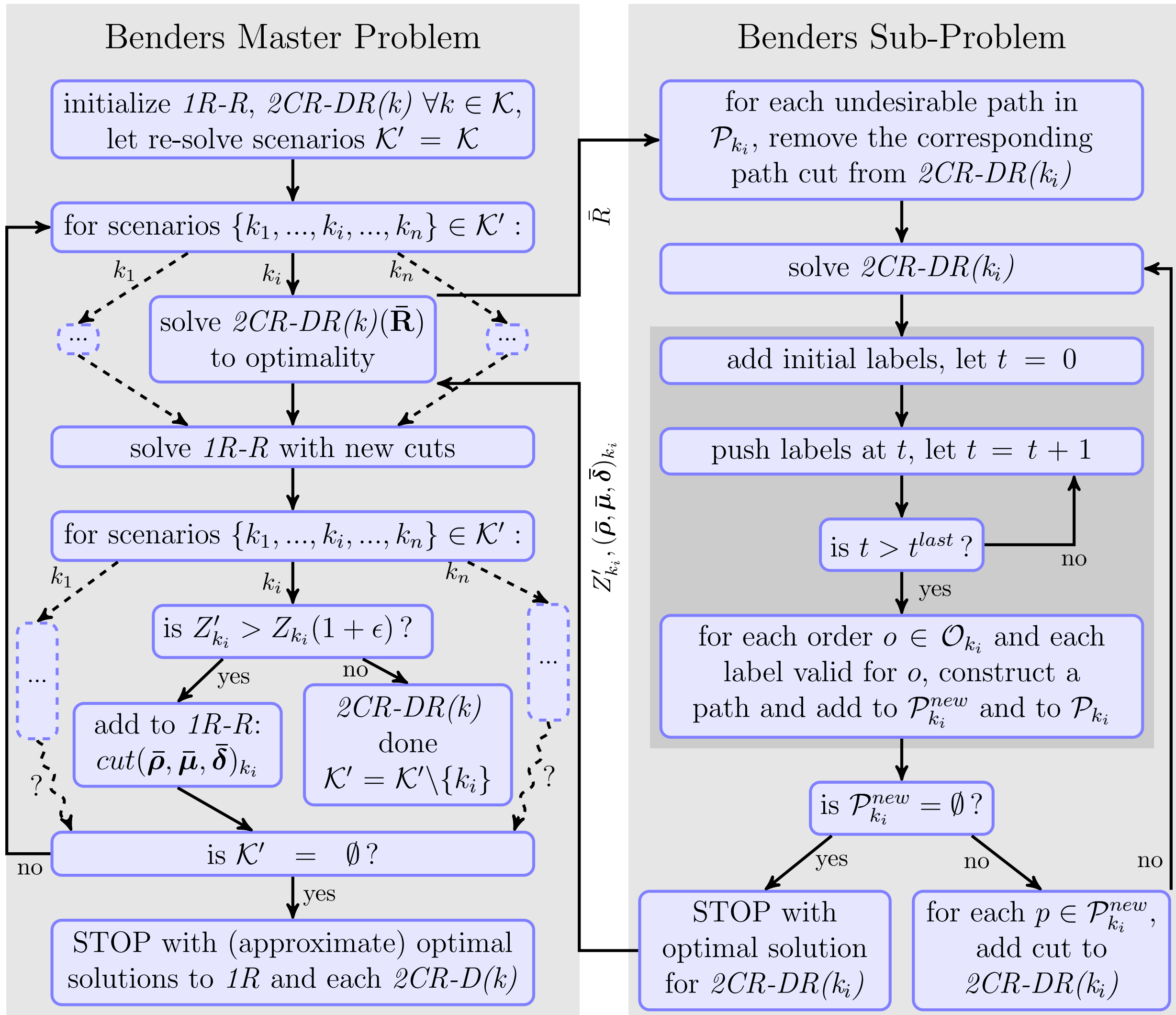
case. Furthermore, the warehouse refrigeration variables link all $|\mathcal{K}|$ second stage commodity routing problems *2CR(k)* with the first stage refrigeration problem *1R* in a set of $|\mathcal{K}| \times |\mathcal{W}|$ hard refrigeration constraints. The full model *RCR* is therefore very likely to be intractable to solve except for a small network G and few scenarios.

3.2 Benders Decomposition for the *RCR*

To overcome the hardness of the linking constraints, a Benders decomposition [9] is used to separate the *1R* and the *2CR(k)* for all scenarios k. Tsitsiklis [49] describes the process of decomposing a general two-stage stochastic problem and using a cutting planes procedure to iteratively add Benders cuts to the relaxed first stage problem until a desired optimality gap is satisfied. A Benders cut is generated from an optimal dual solution to a second stage problem. This Benders framework is adapted for the two-stage stochastic problem *RCR*. Let *1R-R* be the relaxed first stage refrigeration problem, which is the Benders master problem. For a vector of first stage refrigeration decisions $\bar{\mathbf{R}}$ given by *1R-R*, let *2CR-D(k)*($\bar{\mathbf{R}}$) be the full dual problem of the second stage commodity routing problem for scenario k. This model notation is summarized in Table 5.

The general second stage primal problem *2CR* is constructed from the full stochastic problem *RCR* by considering a single scenario and taking the refrigeration decisions as fixed parameters $\bar{R}_w$ $\forall w \in \mathcal{W}$. The now-fixed cost of refrigeration is removed from the objective

Figure 9: Solution Algorithm for the *RCR*



A high-level representation of the Benders decomposition algorithm with the second stage problem solved via a cutting planes algorithm. The dark shaded box within the right box is a very compact representation of the multilabel constrained shortest path algorithm that is used to price paths for the second stage cutting planes procedure. This particular representation uses the rules in Tsitsiklis [49] for choosing which Benders subproblems to re-solve, whereas the representation in Algorithm 3.1 generalizes these rules.

function, and the refrigeration constraints are rearranged so that the parameters appear on the right hand side. Dual prices are then defined on each constraint in the primal problem *2CR*. ρ_w is the dual price on the refrigeration constraint for warehouse $w \in \mathcal{W}$, μ_e is the dual price on the edge flow capacity constraint for edge $e \in \mathcal{E}$, and δ_o is the dual price on the demand constraint for order $o \in \mathcal{O}$.

Model *2CR*: Full Second Stage Commodity Routing Primal Problem

$$
\begin{array}{llllll}
\min & \sum_{p\in\mathcal{P}} c^f_p F_p + \sum_{o\in\mathcal{O}} c^u_o U_o & & & & \text{duals} \\
\text{s.t.} & \sum_{e\in\mathcal{E}(w)} \sum_{p\in\mathcal{P}(e)} r_{p,e} F_p & & \le \hat{m}_w \bar{R}_w & \forall w \in \mathcal{W} & \rho_w \le 0 \\
& \sum_{p\in\mathcal{P}(e)} F_p & & \le m_e & \forall e \in \mathcal{E} & \mu_e \le 0 \\
& \sum_{p\in\mathcal{P}(o)} F_p \; + & U_o & = d_o & \forall o \in \mathcal{O} & \delta_o \ge 0
\end{array}
$$

As in the *RCR*, the refrigeration constraints in *2CR* ensure that for every warehouse $w \in \mathcal{W}$, the total flow over all inventory edges e at warehouse w on paths p for which $r_{p,e} = 1$ (i.e. p assumes that w is refrigerated on inventory edge e) is 0 if w is not refrigerated. The capacity and demand constraints have not changed from the *RCR* formulation.

The dual problem *2CR-D* of *2CR* is now formulated. The columns of *2CR-D*, like the constraints of the primal, correspond to warehouses, edges, and orders. *2CR-D* has two sets of constraints, one indexed over the set of paths $\mathcal{P}$ and one indexed over the set of orders $\mathcal{O}$. The dual prices on the path constraints correspond to the primal path flow variables $F_p \; \forall p \in \mathcal{P}$, and the dual prices on the order constraints correspond to the primal unmet demand variables $U_o \; \forall o \in \mathcal{O}$.

Model *2CR-D*: Full Second Stage Commodity Routing Dual Problem

$$
\begin{array}{llllll}
\max & \sum_{w\in\mathcal{W}} \hat{m}_w \bar{R}_w \rho_w + \sum_{e\in\mathcal{E}} m_e \mu_e + \sum_{o\in\mathcal{O}} d_o \delta_o & & & & \text{duals} \\
\text{s.t.} & \sum_{e\in\mathcal{E}(p)} (r_{p,e} \rho_{w(e)} + \mu_e) + & \delta_{o(p)} & \le c^f_p & \forall p \in \mathcal{P} & F_p \ge 0 \\
& & \delta_o & \le c^u_o & \forall o \in \mathcal{O} & U_o \ge 0
\end{array}
$$

Let D represent the feasible region of *2CR-D*. Let P_D be the set of extreme points of D and let Q_D be the set of extreme rays of D. Since the fixed refrigeration decisions $\bar{\mathbf{R}}$ appear only in the objective function and not in the constraints of *2CR-D*, D does not depend on the refrigeration decisions. To prove that Benders extreme point cuts can always be generated and no Benders feasibility cuts need to be generated in the Benders decomposition algorithm for RCR, it is proved that D is always non-empty and bounded, and therefore contains extreme points but no extreme rays.

Theorem 3.1. (D contains no extreme rays)
The feasible region D of 2CR-D is non-empty and bounded.

Proof. $0 \leq \delta_o \leq c_o^u \; \forall o \in \mathcal{O}$, so let $\delta_o = 0 \; \forall o \in \mathcal{O}$. Substituting into the constraint indexed over paths:

$$\sum_{e \in \mathcal{E}(p)} (r_{p,e}\rho_{w(e)} \; + \; \mu_e) \; \leq c_p^f \;\; \forall p \in \mathcal{P}$$

This inequality is always satisfied because the left hand side is nonpositive ($\rho_w \leq 0 \; \forall w \in \mathcal{W}$ and $\mu_e \leq 0 \; \forall e \in \mathcal{E}$) and $c_p^f \geq 0 \; \forall p \in \mathcal{P}$. Therefore, D is non-empty.

By strong duality theory [49], $D \neq \emptyset$ implies that the primal problem *2CR* is either infeasible or feasible and bounded. For any vector $\bar{\mathbf{R}}$, a feasible solution for *2CR*($\bar{\mathbf{R}}$) is $(\mathbf{F}, \mathbf{U}) = (\mathbf{0}, \mathbf{d})$, so *2CR* is never infeasible. This implies that the primal-dual pair is feasible and bounded. Therefore, D is non-empty and bounded. $\square$

By Theorem 3.1, D is non-empty and contains no extreme rays ($Q_D = \emptyset$). Therefore, the optimal objective value of *2CR* and *2CR-D* can be characterized in terms of only the extreme points P_D of *2CR-D*. For a vector $\bar{\mathbf{R}}$ of refrigeration decisions, let the optimal objective value of *2CR-D*($\bar{\mathbf{R}}$) be $Z'(\bar{\mathbf{R}})$.

$$Z'(\bar{\mathbf{R}}) \;\; = \;\; \max_{(\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\delta}) \in P_D} \left(\sum_{w \in \mathcal{W}} \hat{m}_w \bar{R}_w \rho_w \;\; + \; \sum_{e \in \mathcal{E}} m_e \mu_e \;\; + \; \sum_{o \in \mathcal{O}} d_o \delta_o \right)$$

Alternatively, $Z'(\bar{\mathbf{R}})$ is the smallest number Z' such that the following inequalities hold:

$$Z' \;\; \geq \; \sum_{w \in \mathcal{W}} \hat{m}_w \bar{R}_w \rho_w \;\; + \; \sum_{e \in \mathcal{E}} m_e \mu_e \;\; + \; \sum_{o \in \mathcal{O}} d_o \delta_o \qquad \forall (\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\delta}) \in P_D$$

For some vector $\bar{\mathbf{R}}$ and scenario $k \in \mathcal{K}$, let Z'_k be the optimal value for the second stage problem *2CR-D(k)*($\bar{\mathbf{R}}$) with feasible region D_k and extreme points P_{D_k}. It is now possible to formulate the first stage refrigeration problem *1R* as a Benders master problem.

Model *1R*: First Stage Refrigeration Problem

$$\begin{aligned} \min \quad & \sum_{k\in\mathcal{K}} \alpha_k Z_k \;+\; \sum_{w\in\mathcal{W}} c^r_w R_w \\ \text{s.t.} \quad & Z_k \;\geq\; \sum_{w\in\mathcal{W}} \hat{m}_{k,w}\bar{\rho}_w R_w \;+\; \sum_{e\in\mathcal{E}} m_{k,e}\bar{\mu}_e \;+\; \sum_{o\in\mathcal{O}_k} d_o\bar{\delta}_o \qquad \forall k\in K,\; \left(\bar{\boldsymbol{\rho}},\bar{\boldsymbol{\mu}},\bar{\boldsymbol{\delta}}\right)\in P_{D_k} \end{aligned}$$

This formulation for *1R* reduces the number of variables substantially, from $|\mathcal{W}|+\sum_{k\in\mathcal{K}}|\mathcal{P}_k|+\sum_{k\in\mathcal{K}}|\mathcal{O}_k|$ in the full *RCR* formulation to just $|\mathcal{K}|+|\mathcal{W}|+|\mathcal{E}|+\sum_{k\in\mathcal{K}}|\mathcal{O}_k|$ in the Benders master problem *1R*. However, the number of constraints is equal to the number of extreme points in D_k, and the number of extreme points is usually exponential in the number of variables and constraints [49]. However, most of these constraints are likely to be inactive at an optimal solution, so a cutting planes algorithm may be used to solve *1R*. In each iteration of the cutting planes algorithm, a relaxed problem *1R-R* is solved and cuts violating the current solution are added to *1R-R* [49]. Maintaining a much smaller set of constraints is likely to make *1R* more tractable to solve.

3.2.1 Cut Generation Algorithm for the Benders Decomposition

In one iteration of the cutting planes algorithm for the Benders master problem *1R*, the relaxed master problem *1R-R* is solved and one or more violated Benders cuts are added. A violated constraint is identified by an optimal extreme point solution $\left(\bar{\boldsymbol{\rho}},\bar{\boldsymbol{\mu}},\bar{\boldsymbol{\delta}}\right)\in P_{D_k}$ of the second stage dual problem *2CR-D(k)* if the optimal cost Z'_k of the solution is greater than Z^*_k, the variable in *1R-R* representing the maximum second stage cost for scenario $k\in\mathcal{K}$.

If $Z'_k > Z^*_k$, the cutting planes algorithm adds the following violated cut to the relaxed master problem *1R-R*.

$$Z_k \;\geq\; \sum_{w\in\mathcal{W}} \hat{m}_{k,w}\bar{\rho}_w R_w \;+\; \sum_{e\in\mathcal{E}} m_{k,e}\bar{\mu}_e \;+\; \sum_{o\in\mathcal{O}_k} d_o\bar{\delta}_o$$

Algorithm 3.1 describes the Benders decomposition solution method for the *RCR* problem. The algorithm requires a fully relaxed first stage refrigeration problem *1R-R* with no cuts, a set $\{$*2CR-D(k)*$: k\in\mathcal{K}\}$ of second stage dual problems, an initial re-solve set $\mathcal{K}^r$ of scenarios $k\in\mathcal{K}$ to solve in the first master iteration, and a desired relative optimality gap ϵ. After each second stage problem in the re-solve set $\mathcal{K}^r$ is solved to optimality from its previous optimal basis (Line 6), a (sufficiently) violating cut is added to *1R-R* if it exists. The updated MIP *1R-R* is then solved to optimality from its previous optimal basis (Line 11). According to some specified rule, the function Re-solveSet then selects a subset $\mathcal{K}^r\subseteq\mathcal{K}$ to be the set of second stage problems to be re-solved at the next iteration (Line 12). The Benders algorithm is terminated when the Re-solveSet function returns no subproblems to be re-solved, and if this only occurs when $Z'_k \leq Z^*_k(1+\epsilon)\ \forall k \in \mathcal{K}$, then a near-optimal solution is returned for small ϵ (or an exact optimal solution if $\epsilon = 0$).

Algorithm 3.1 Benders Decomposition Algorithm for First Stage Refrigeration Problem

procedure $\textsc{Benders}(\mathit{1R\text{-}R}, \{\mathit{2CR\text{-}D(k)} : k \in \mathcal{K}\}, \mathcal{K}, \mathcal{K}^r, \epsilon)$
  $P_{\mathit{1R\text{-}R}} \leftarrow \{(\mathbf{R}, \mathbf{Z}) : R_w \in \{0,1\}\ \forall w \in \mathcal{W}, Z_k \geq 0\ \forall k \in \mathcal{K}\}$ ▷ Begin with no cuts
  $(\mathbf{R}^*, \mathbf{Z}^*) \leftarrow (\mathbf{0}, \mathbf{0})$ ▷ Begin with trivial solution
  while $\mathcal{K}^r \neq \emptyset$ **do** ▷ Loop while ∃ sub-problems to re-solve
    for all $k \in \mathcal{K}^r$ **do** ▷ For each sub-problem to re-solve
      $(Z'_k, (\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\delta}})_k) \leftarrow \textsc{Solve}(\mathit{2CR\text{-}D(k)}(\mathbf{R})$ ▷ Solve the sub-problem
      if $Z'_k > Z^*_k(1+\epsilon)$ **then** ▷ Violating cut exists
        $P_{\mathit{1R\text{-}R}} \leftarrow P_{\mathit{1R\text{-}R}} \cap \{(\mathbf{R}, \mathbf{Z}) : Z_k \geq \hat{\mathbf{m}}_{\mathbf{k}} \bar{\boldsymbol{\rho}} \mathbf{R} + \mathbf{m}_{\mathbf{k}} \bar{\boldsymbol{\mu}} + \mathbf{d} \bar{\boldsymbol{\delta}}_k\}$ ▷ Add cut
      end if
    end for
    $(\mathbf{R}^*, \mathbf{Z}^*) \leftarrow \textsc{Solve}(\mathit{1R\text{-}R})$ ▷ Solve restricted master
    $\mathcal{K}^r \leftarrow \textsc{Re-solveSet}(\mathcal{K}, \mathcal{K}^r, \mathbf{Z}^*, \mathbf{Z}', \dots)$ ▷ Identify sub-problems to re-solve
  end while
  return $\mathbf{R}^*$
end procedure

Tsitsiklis [49] suggests re-solving each second stage problem *2CR-D(k)* for which $Z'_k > Z^*_k(1+\epsilon)$ prior to solving the *1R-R* again. However, this is not necessarily the best approach for every problem, as demonstrated by Tsamasphyrou et al. [48] in their Benders algorithm for a stochastic power generation model. When the Benders master problem is relatively easy to solve in comparison to a Benders subproblem, solving fewer subproblems between each master solve may accelerate the convergence of the Benders algorithm to an optimal solution. This is because introducing cuts more regularly means the refrigeration decisions $\bar{\mathbf{R}}$ are updated with new information from the subproblems more quickly, which is likely to reduce the total number of subproblem iterations needed. In the computational testing section of this paper, several different rules for choosing the re-solve set $\mathcal{K}^r \subseteq \mathcal{K}$ are tested.

3.3 Cutting Planes Algorithm for a *2CR-D*

The general second stage commodity routing problem *2CR* is a multicommodity flow problem with side constraints on paths. This class of problems is known to be NP-hard with the constrained shortest-path problem as a special case [28]. Since the number of paths $|\mathcal{P}_k|$ in a second stage problem *2CR(k)* is exponential in the size of the graph G, the dual problem *2CR-D(k)* has a huge number of constraints (equivalently, *2CR(k)* has a huge number of variables) and is therefore likely to be intractable.

Fortunately, a cutting planes approach can be used to solve each *2CR-D(k)* (equivalently, a column generation approach for *2CR(k)*) without having to enumerate all paths in G. Re-define $\mathcal{P}_k$ as the current restricted set of paths for scenario $k \in \mathcal{K}$, and let *2CR-DR(k)* be the relaxed second stage problem with path constraints indexed over $\mathcal{P}_k$. At the start of the cutting planes algorithm, *2CR-DR(k)* is initialized with $\mathcal{P}_k = \emptyset$.

During an iteration of the Benders algorithm, the relaxed Benders master problem is solved and an updated vector of refrigeration decisions $\bar{\mathbf{R}}$ is passed to *2CR-DR(k)* if k is in the re-solve set $\mathcal{K}^r$. Since the refrigeration decisions $\bar{\mathbf{R}}$ do not appear in any of the constraints of *2CR-DR(k)*, $\bar{\mathbf{R}}$ does not impact the feasible region of *2CR-DR(k)*. $\bar{\mathbf{R}}$ appears as a parameter in the objective function of *2CR-DR(k)*, so a change in $\bar{\mathbf{R}}$ following a Benders iteration may mean the current basis is no longer optimal for the full dual problem *2CR-D(k)*. Starting from this basis, *2CR-DR(k)*$(\bar{\mathbf{R}})$ is re-solved to optimality with the restricted set of path cuts. Since the Benders algorithm needs an optimal extreme point solution to the full dual problem *2CR-D(k)*, the second stage cutting planes algorithm must iteratively add newly-violating path cuts to the restricted problem *2CR-DR(k)*$(\bar{\mathbf{R}})$ until a new optimal solution $(\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\delta}) \in P_{D_k}$ for *2CR-D(k)* is found.

Algorithm 3.2 Cutting Planes Algorithm for Second Stage Commodity Routing Problem

1: **procedure** CuttingPlanes(*2CR-DR(k)*, $\mathcal{P}_k$, $\bar{\mathbf{R}}$)
2: *2CR-DR(k)* $\leftarrow$ UpdateObj(*2CR-DR(k)*, $\bar{\mathbf{R}}$) ▷ Update $\bar{\mathbf{R}}$ parameters in objective
3: (*2CR-DR(k)*, $\mathcal{P}_k$) $\leftarrow$ InitDrop(*2CR-DR(k)*, $\mathcal{P}_k$) ▷ Drop unwanted existing cuts
4: $\mathcal{P}_k^{new} \leftarrow \emptyset$ ▷ Initialize set of new paths
5: **repeat**
6: (*2CR-DR(k)*, $\mathcal{P}_k$) $\leftarrow$ LoopDrop(*2CR-DR(k)*, $\mathcal{P}_k$) ▷ Drop unwanted cuts
7: **for all** $p \in \mathcal{P}_k^{new}$ **do** ▷ For each new path
8: $P_{\textit{2CR-DR(k)}} \leftarrow P_{\textit{2CR-DR(k)}} \cap \left\{ (\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\delta}})_k : \sum_{e \in \mathcal{E}(p)} (r_{p,e}\rho_{w(e)} + \mu_e) + \delta_{o(p)} \leq c_p^f \right\}$ ▷ Add cut
9: **end for**
10: $\mathcal{P}_k \leftarrow \mathcal{P}_k \cup \mathcal{P}_k^{new}$ ▷ Add new paths to existing path set
11: $(Z_k', (\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\delta}})_k) \leftarrow$ Solve(*2CR-DR(k)*) ▷ Solve relaxed second stage problem
12: $\mathcal{P}_k^{new} \leftarrow$ FindPaths($(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\delta}})_k, \mathcal{P}_k$) ▷ Find a set of good new paths
13: **until** $\mathcal{P}_k^{new} = \emptyset$ ▷ End loop when no new paths
14: **return** $(Z_k', (\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\delta}})_k, \mathcal{P}_k)$
15: **end procedure**

Algorithm 3.2 describes the cutting planes algorithm for a second stage commodity routing problem, which requires a problem *2CR-DR(k)* and a previous initial basis, the restricted set of paths from the previous solve, and a vector of refrigeration decisions from the Benders master problem. The InitDrop function (Line 3) uses a specified rule to relax a subset of the path cuts that were present in *2CR-DR(k)* at the end of a previous call to Algorithm 3.2 and before a change in $\bar{\mathbf{R}}$. The LoopDrop function (Line 6) performs the same task during iterations of the second stage cutting planes algorithm, and may use a different rule for choosing path cuts to relax. Dropping constraints with high slack, or equivalently, dropping variables with large positive reduced cost, is a common technique that can improve the efficiency of large-scale optimization algorithms by keeping the problem size small. In the computational testing section of this paper, several combinations of rules for dropping path cuts are tested.

After each solve of the relaxed second stage problem *2CR-DR(k)* (Line 11), the FindPaths procedure is passed the current solution and the restricted set of paths (Line 12). This procedure is the path pricing algorithm, described in Section 3.4. FindPaths returns a set of new paths $\mathcal{P}_k^{new}$ that are feasible for the implicit shelf-life expenditure constraints and correspond to violating cuts for *2CR-DR(k)*. If $\mathcal{P}_k^{new} \neq \emptyset$, a cut is added to *2CR-DR(k)* for each $p \in \mathcal{P}_k^{new}$ (Line 8) and the new paths are added to the set of existing paths $\mathcal{P}_k$ (Line 10). Algorithm 3.2 terminates with an optimal solution for the full second stage dual problem *2CR-D(k)*($\bar{\mathbf{R}}$) when no new feasible paths corresponding to violating cuts are generated by FindPaths.

3.4 Path Pricing Algorithm for the Second Stage Cutting Planes Procedure

Given a current optimal solution $(\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\delta})$ for a relaxed second stage problem *2CR-DR*, the path pricing algorithm must generate paths that correspond to violated cuts and are feasible for the shelf-life constraints. A path p was re-defined as an ordered list of pairs $(e, r_{p,e})$, where $e \in \mathcal{E}$ is an edge traversed by the p and $r_{p,e} \in \{0,1\}$ indicates whether edge e is an inventory edge associated with a warehouse that p assumes to be refrigerated. The edges in this ordered list of pairs must be traversable from the supersource *src* to a destination node (a time-expanded warehouse node) at which demand for order $o(p)$ can be satisfied. If $t_{o(p)}^{late} \in \mathbb{Z}^+$ is the number of allowable late periods for the order $o(p) = (g, w, t)$, the destination node must be one of $(w,t), (w,t+1), ..., (w, t+t_o^{late}-1), (w, t+t_o^{late})$.

Thus the pricing subproblem for the second stage cutting planes algorithm can be thought of as an optimization problem that seeks to find a traversable list of such pairs satisfing an order o with a minimal overall reduced cost, subject to the constraint that the sum of shelf-life expenditures on these edge pairs is no greater than l_o^{max}. If an optimal (feasible) path exists for this optimization problem, it shall be returned to the cutting planes master problem *2CR-DR(k)* if its cost is negative. If the optimal objective value is nonnegative, no paths with negative reduced cost exist. In this case, there are no more violated cuts to add to *2CR-DR(k)*, so the current solution $(\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\delta}) \in P_{D_k}$ is optimal for the full dual second stage problem *2CR-D(k)*($\bar{\mathbf{R}}$).

3.4.1 Minimizing the Reduced Costs of Paths

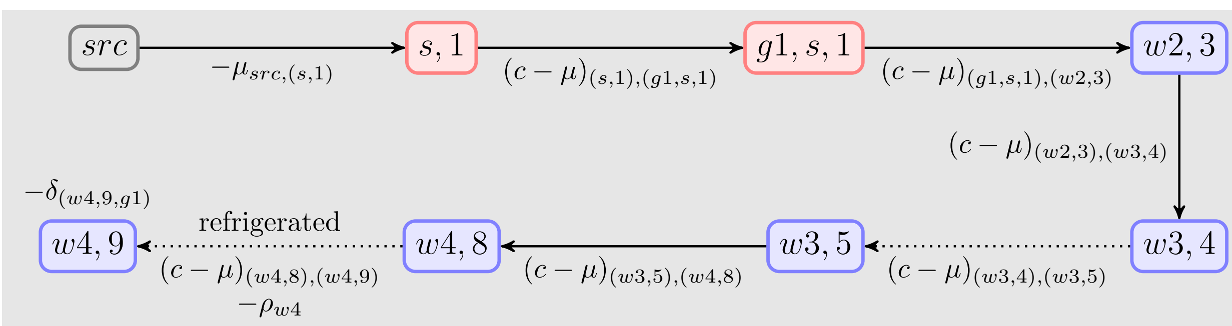
The cost of a path p is the sum of the costs of the edges on the path plus the lateness cost for the path. The lateness cost for a path satisfying order $o(p) = (w, t, g)$ with destination node $(w, t+\tau), \tau \in \{1, \dots, t_{o(p)}^{late}\}$ is the cost per late period times the number of late periods, $\tau \times c_{o(p)}^{late}$. So for a path p that delivers goods τ periods late for order $o(p)$, the cost of p is $c_p^f = \sum_{e \in \mathcal{E}(p)} c_e + \tau c_{o(p)}^{late}$. Given a current optimal solution $(\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\delta})$ for the relaxed dual problem, the negative reduced cost condition for a path p can be rearranged and written as follows.

$$\sum_{e \in \mathcal{E}(p)} (c_e - r_{p,e}\rho_{w(e)} - \mu_e) \;-\; \delta_{o(p)} + \tau c_{o(p)}^{late} \;< 0$$

For the general problem of finding the lowest reduced cost path on a network, the typical approach is to use dual solution values to modify the costs on nodes and edges such that an efficient shortest-path type algorithm will minimize reduced cost [36]. The sum in the reduced cost inequality is indexed over edges $e \in \mathcal{E}(p)$ for path p, so each edge in G is assigned the modified cost $c_e - r_{p,e}\rho_{w(e)} - \mu_e$ for a path p. For any noninventory edge e, $r_{p,e} = 0$ for all p, and for an inventory edge e, $r_{p,e} = 1$ if e is assumed to be refrigerated on p. Therefore, this edge cost assignment has the desired effect of subtracting the dual variables $\rho_{w(e)}$ for all inventory edges e on a path that are assumed to be refrigerated. The reduced cost equation also adds $-\delta_{o(p)} + \tau c_{o(p)}^{late}$ to the cost of path p, so for $o = (w, t, g)$, so each node $(w, t + \tau), \tau \in \{0, 1, \ldots, t^{late}\}$ is assigned this term. Now, a shortest path in the graph G containing these modified edge and node costs will be the smallest reduced cost path.

Figure 10 offers a simple example of a path on the graph with modified edge and node costs. Note that the costs on edges are always nonnegative because of the signs on the dual variables ($\mu_e \le 0 \; \forall e \in \mathcal{E}$ and $\rho_w \le 0 \; \forall w \in \mathcal{W}$) and the fact that edge flow costs c_e are nonnegative.

Figure 10: An Example Path, Displaying Modified Node and Edge Costs



An example path valid for order $(w4, 9, g1)$, with modified edge costs taking dual values from *2CR-DR*. The path delivers goods on time since the destination node $(w4, 9)$ coincides with the order node. Inventory edges are dotted, and an inventory edge that is assumed to be refrigerated on the path is labeled.

The path in Figure 10 leaves good-supplier node $(g1, s, 1)$, and passes over transport edges $((g1, s, 1), (w2, 3))$, $((w2, 3), (w3, 4))$, $((w3, 5), (w4, 8))$, an inventory edge $((w3, 4), (w3, 5))$ assumed to be nonrefrigerated, and an inventory edge $((w4, 8), (w4, 9))$ that is assumed to be refrigerated. The path ends at a node corresponding to the order $(w4, 9, g1)$, so units of $g1$ delivered on this path will fulfill the order on-time. If instead the path ended at a node that is late for the order, the lateness penalty would be subtracted from the cost $-\delta_o$ on the destination node. The reduced cost $rc(p)$ of the path p in Figure 10 is therefore the sum of the edge costs and the node cost $-\delta_{(w4,9,g1)}$ at $(w4, 9)$.

$$\begin{aligned} rc(p) = &\ -\mu_{src,(s,1)} + (c-\mu)_{(s,1),(g1,s,1)} + (c-\mu)_{(g1,s,1),(w2,3)} + (c-\mu)_{(w2,3),(w3,4)} \\ &+(c-\mu)_{(w3,4),(w3,5)} + (c-\mu)_{(w3,5),(w4,8)} + (c-\mu)_{(w4,8),(w4,9)} - \rho_{w4} - \delta_{(w4,9,g1)} \\ = &\ -\rho_{w4} + \sum_{e\in\mathcal{E}(p)} (c_e - \mu_e) - \delta_{(w4,9,g1)} \\ = &\ \sum_{e\in\mathcal{E}(p)} (c_e - r_{p,e}\rho_{w(e)} - \mu_e) - \delta_o + c_o^{late}\tau \qquad \tau = 0 \text{ since } p \text{ arrives on time} \end{aligned}$$

If this reduced cost is negative and the path is feasible for the shelf-life expenditure constraint, then the pricing problem may generate this path so that a violated cut can be added to *2CR-DR*. Note that on the inventory edge that the path displayed traverses, the dual value $\rho_{w4} \leq 0$ on the refrigeration constraint for warehouse $w4$ appears in the edge cost. This makes intuitive sense: if warehouse $w4$ is currently not refrigerated (i.e. $\bar{R}_{w4} = 0$ in the first stage decision vector $\bar{\mathbf{R}}$), then if the refrigeration constraint is currently tight $\rho_{w4} < 0$, subtracting this dual value in the reduced cost calculation will increase the reduced cost of the path. This makes it less likely that the pricing subproblem will generate the path.

3.4.2 Satisfying the Shelf-life Expenditure Constraints

Figure 11 illustrates the accumulation of shelf-life expenditure over edges in an example path in G. A simple shortest-path type algorithm may return a path that is infeasible for the implicit shelf-life expenditure constraint. This constraint limits the total shelf-life that can be expended on all edges traversed by p to no greater $l_{o(p)}^{max}$, the allowable shelf-life parameter for the order that the path satisfies, so it can be written as follows. Note that l_e^r was not defined on non-inventory edges e, but since $r_{p,e} = 0$ for such edges, l_e^r can be made arbitrary for non-inventory edges.

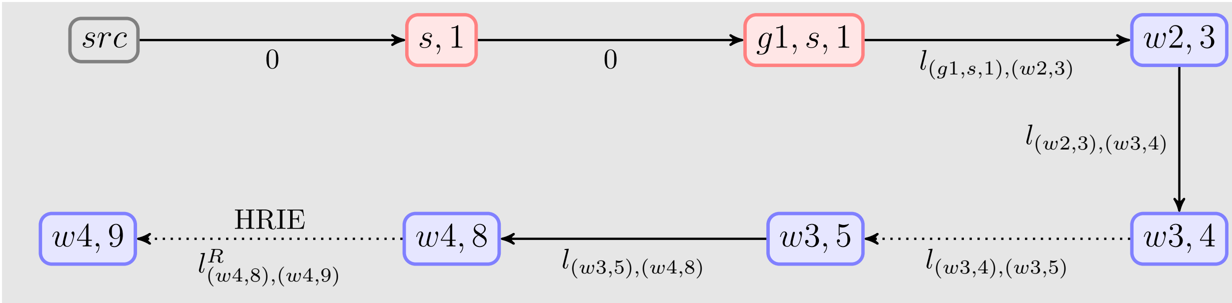
$$\sum_{e\in\mathcal{E}(p)} (l_e - l_e^r r_{p,e}) \ < l_{o(p)}^{max}$$

3.4.3 A Multilabel Algorithm for the Path Pricing Subproblem

The purpose of the second stage pricing problem is to avoid the enumeration of paths in the network, which is very expensive computationally and in terms of memory. Multilabel shortest path algorithms use a concept of path domination to ensure that the only nondominated paths are returned while avoiding enumeration of all paths. The path pricing problem has two 'resources' on paths: a reduced cost and a remaining shelf-life expenditure.

Algorithms 3.3 to 3.7 describe the multilabel constrained shortest path algorithm. Algorithm 3.3 (FindPaths) is the full algorithm, which calls Algorithms 3.4 to 3.5 (LabelsPush)

Figure 11: An Example Path, Displaying Shelf-Life Expenditures on Edges



An example path valid for order $(w4, 9, g1)$, with fixed shelf-life expenditures on edges.

in the main loop and then Algorithm 3.7 (ConstructPaths) once right before terminating. FindPaths requires a current vector of dual variables optimal for the relaxed second stage dual problem *2CR-DR(k)* and the current restricted set of paths. Several dictionaries, defined in Table 9 are initialized (Line 2).

Table 9: Data Structures and Parameters

IL	dict of tuples	maps from unique label index n to contents of the label
NI	dict of lists	maps from node to ordered list of label indices at the node
TN	dict of sets	maps from time step to set of nodes at the time step
MCL	dict of dicts	maps from a node to orders reachable from the node and from an order at a node to a list of (cost, shelf life) pairs for all nondominated path extensions to the order
n	counter	unique label index, a counter for the number of labels added
pls	list	potential label tuples that have not yet been checked or added
pl	tuple	a potential label (pred index, current node, order, reduced cost, shelf-life remaining, status), where status is $True$ iff the last edge was an inventory edge assumed refrigerated, else $False$
pos	counter	the current position in a list of existing labels

A loop then adds a set of initial labels for each order to the good-supplier nodes (Lines 4-18). These initial labels are assigned cumulative reduced costs (rc) equal to the sum of the edge costs on aggregate and individual good supply edges (Line 7, Line 11), and remaining shelf-lives (lr) equal to the parameter for the maximum shelf-life expenditure on a path for order o, l_o^{max} (Line 8, Line 11). The bounding conditions are then checked (Lines 9-10) and if the order cannot be reached from the good-supplier node, then no initial label is added at the node. If the order is reachable, a potential label is instantiated in a list and the AddNondomLabels function is called on the list to add the label to the good-supply node if it is not dominated by any existing labels at the node (Line 12). The node is then added to the set of nodes at time t that have labels (Line 13).

The main loop for the multilabel algorithm (Lines 19-29) is initialized with a current time

Algorithm 3.3 Multilabel Constrained Shortest Paths Algorithm

1: **procedure** FindPaths($(\bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\delta}})_k, \mathcal{P}_k$)
2: $IL \leftarrow dict(), NI \leftarrow dict(), TN \leftarrow dict()$ ▷ Initialize dicts
3: $n \leftarrow 1$ ▷ Begin with current label index 1
Add initial labels at good-supplier nodes
4: **for all** $(g, s, t) \in G$ **do** ▷ For each good-supplier node
5: **for all** $(w, t', g') \in \mathcal{O}_k$ **do** ▷ For each order
6: **if** $g = g' \land (g, s, t) \in MCL \land (w, t', g') \in MCL(g, s, t)$ **then** ▷ If order and node goods match and feasible path is not impossible from node to order
7: $rc \leftarrow -\delta_{(w,t',g')} + c_{(\text{`}src\text{'},(s,t))} - \mu_{(\text{`}src\text{'},(s,t))} + c_{((s,t),(g,s,t))} - \mu_{((s,t),(g,s,t))}$ ▷ Calc. reduced cost at node
8: $lr \leftarrow l^{max}_{(w,t',g')}$ ▷ Life remaining is max. path expenditure
9: **for all** $(mc, ml) \in MCL(g, s, t)(w, t', g')$ **do** ▷ For each min(cost,life) pair
10: **if** $mc < -rc(1 - \epsilon') \land lr \geq ml$ **then** ▷ If bounds are satisfied
11: $pls \leftarrow list((0, (g, s, t), (w, t', g'), rc, lr, False))$ ▷ Create pot. label
12: $n \leftarrow$ AddNondomLabels($n, (g, s, t), pls$) ▷ Add nondom. label
13: $TN(t) \leftarrow TN(t) \cup \{(g, s, t)\}$ ▷ Add node to set of nodes at time
14: **end if**
15: **end for**
16: **end if**
17: **end for**
18: **end for**
Push labels forward in G from nodes at each sequential time step, checking dominance
19: $t^{curr} \leftarrow 0$ ▷ Initialize current time step to first
20: **while** $t^{curr} \leq |\mathcal{T}| - 1$ **do** ▷ Loop until current time step equals final time step
21: **for all** $i \in TN(t^{curr})$ **do** ▷ For each node at current time step
22: **for all** $(\_, j) \in OutEdges_G(i)$ **do** ▷ For each edge leaving the node
23: **if** $j \in MCL$ **then** ▷ If to-node can reach any orders
24: $n \leftarrow$ LabelsPush(n, i, j) ▷ Push labels to to-node
25: **end if**
26: **end for**
27: **end for**
28: $t^{curr} \leftarrow t^{curr} + 1$ ▷ Repeat for next time step
29: **end while**
30: **return** ConstructPaths($IL, NI, \mathcal{P}_k$) ▷ Construct unique paths
31: **end procedure**

step t^{curr} of 0. The main loop iterates by incrementing t^{curr} by 1 (Line 28), moving from one time step to the next, until there are no more time steps in the time horizon $\mathcal{T}$. In a single iteration, for each good-supplier node and time-expanded warehouse node i at time step t^{curr}, and for each edge (i, j) leaving node i, if any orders are reachable from node j

then the LabelsPush function is called to push the labels existing at node i over edge (i, j) to node j (Line 24). Since G has a special directed acyclic structure for which all edges from a good-supplier or time-expanded warehouse node point to nodes at strictly later time steps, when this main loop terminates, all possible labels have been 'fixed', meaning all feasible paths with negative reduced costs are encoded by the final set of labels stored in dictionary IL. The multilabel algorithm in Algorithm 3.3 then calls the ConstructPaths function to create the paths from these fixed labels (Line 30), and the set of new paths that is returned is given directly to the cutting planes algorithm that called the multilabel algorithm.

The LabelsPush function of the multilabel algorithm is described in Algorithms 3.4 to 3.5. This function requires the current last label index (n), the node that labels are being pushed from (i) and the node that labels are to be pushed to (j). LabelsPush first checks whether the edge (i, j) is an inventory edge or a transport edge. If (i, j) is an inventory edge, labels at node i are taken one at a time and pushed over the inventory edge twice, once under the assumption that the edge is not refrigerated (Lines 5-12), and once assuming it is refrigerated (Lines 13-21). For each of these two cases, the reduced cost and remaining shelf-life of the pushed label is calculated (Lines 5-6, 13-14) and the bounding conditions are checked. If the order $o(n')$ corresponding to the label n' is reachable from node j, i.e. if it is not the case that there are no extensions of the path at node j to a node satisfying order $o(n')$ that will result in a feasible path, then the pushed label is instantiated and appended to the end of the current list of potential labels $Npls$ (potential labels pushed over an inventory edge assumed not refrigerated) or $Rpls$ (potential labels pushed over an inventory edge assumed refrigerated). When this loop is complete for all labels n' at node i, the function AddNondomLabels is called once for each list of potential labels $Npls$ and $Rpls$ (Lines 22-27). If edge (i, j) is instead a transport edge, then an equivalent procedure is conducted except only once for each label (Lines 29-43), since there are no refrigeration assumptions needed for transport edges. Finally, LabelsPush function returns the last index n that was updated during the AddNondomLabels calls.

The AddNondomLabels function is described in Algorithm 3.6. AddNondomLabels requires the current last label index and the node j that the list of potential labels pls (also required) is being pushed to. It checks the label domination conditions before adding new nondominated labels from pls to the list of existing labels at node j and removing existing labels that are dominated from the list at node j. An invariant is maintained on all lists of labels that ensures the labels are ordered so that the reduced costs are nondecreasing as the list is traversed from left to right. Another invariant maintained on all lists of labels is that no label in a list dominates any other label, or all labels are mutually nondominating. Since the AddNondomLabels function maintains these invariants and potential labels are only ever added to label lists through this function (even the initial labels at good-supplier nodes), the reduced cost ordering and mutual nondomination invariants are never broken and therefore never need to be 'checked' or 'repaired'. These two invariants together imply that the shelf-life remaining for the labels in a list is automatically ordered in nondecreasing manner as well, else there would be domination.

Algorithm 3.4 Labels Push Function, part 1

```
procedure LabelsPush(n, i, j)
    if w(i) = w(j) then                                  ▷ If edge is inventory edge
        Npls ← list(), Rpls ← list()                     ▷ Initialize lists of potential labels
        for all n′ ∈ NI(i) do                            ▷ For each label index at node
Push label over inventory edge assuming no refrigeration
            Nrc ← rc(n′) + c_(i,j) − μ_(i,j)             ▷ Update reduced cost of sub-path
            Nlr ← lr(n′) + l^N_(i,j)                     ▷ Update remaining shelf-life of sub-path
            for all (mc, ml) ∈ MCL(j)(o(n′)) do          ▷ For each min(cost,life)
                if mc < −Nrc(1 − ε′) ∧ Nlr ≥ ml then     ▷ If bounds are satisfied
                    Append(Npls, (n′, j, o(n′), Nrc, Nlr, False))   ▷ Add pot. nonr. label
                    BreakFor
                end if
            end for
Push label over inventory edge assuming refrigeration
            Rrc ← rc(n′) + c_(i,j) − μ_(i,j) − ρw(i)     ▷ Update reduced cost of sub-path
            Rlr ← lr(n′) + l^R_(i,j)                     ▷ Update remaining shelf-life of sub-path
            for all (mc, ml) ∈ MCL(j)(o(n′)) do          ▷ For each min(cost,life)
                if mc < −Rrc(1 − ε′) ∧ Rlr ≥ ml then     ▷ If bounds are satisfied
                    Append(Npls, (n′, j, o(n′), Rrc, Rlr, True))    ▷ Add pot. refr. label
                    BreakFor
                end if
            end for
        end for
Add nondominated labels for order at to-node
        if length(Npls) > 0 then           ▷ If nonrefrigerated potential label list not empty
            n ← AddNondomLabels(n, j, Npls)                ▷ Add nondom. labels at j
        end if
        if length(Rpls) > 0 then           ▷ If refrigerated potential label list not empty
            n ← AddNondomLabels(n, j, Rpls)                ▷ Add nondom. labels at j
        end if
```

Maintaining only nondominated labels makes the overall multilabel algorithm more efficient because dominated labels are not useful in constructing a Pareto optimal set of paths with respect to reduced cost and shelf-life remaining. The ordering on each list of labels makes the AddNondomLabels more efficient because on average (but not in the worst case), it means many fewer pairwise comparisons between potential labels and existing labels at node j are needed to establish whether there is a dominating relationship.

AddNondomLabels is initialized at position $pos = 0$ in the list of existing labels at j. For each label pl in the list of potential labels (which is ordered because the same edge cost and shelf-life expenditure was mapped onto all labels in the list), a set of conditions is

Algorithm 3.5 Labels Push Function, part 2

```
    else                                                    ▷ Edge is transport edge
        pls ← list()                                        ▷ Initialize list of potential labels
        for all n′ ∈ NI(i) do                               ▷ For each label index at node/order
Push label over transport edge
            rc ← rc(n′) + c_(i,j)                           ▷ Update reduced cost of sub-path
            lr ← lr(n′) + l_(i,j)                           ▷ Update remaining shelf-life of sub-path
            for all (mc, ml) ∈ MCL(j)(o(n′)) do             ▷ For each min(cost,life)
                if mc < −rc(1 − ϵ′) ∧ lr ≥ ml then          ▷ If bounds are satisfied
                    Append(pls, (n′, j, o(n′), rc, lr, False))   ▷ Add pot. label
                    BreakFor
                end if
            end for
        end for
Add nondominated labels for order at to-node
        if length(pls) > 0 then                             ▷ If potential label list not empty
            n ← AddNondomLabels(n, j, pls)                  ▷ Add nondom. labels at j
        end if
    end if
    return n
end procedure
```

checked only once. The current position *pos* in the existing label list $NI(j)$ is incremented until the reduced cost of the potential label is less than or equal to the reduced cost of the existing label at *pos* or the end of the list is reached (Lines 4-6). Next, the function checks whether the potential label is dominated by the existing label at the current position or by the existing label at the previous position (Lines 7-9). This is the case if the reduced costs of the labels are equal and the the life remaining of the potential label is smaller, or if the life remaining of the previous position existing label exceeds that of the potential label. If the potential label is dominated, it is discarded, and the next label is pulled off the potential label list. Note that the position is not reset to 0 when this occurs, since the potential labels are ordered by reduced cost. If the potential label is not dominated, then it is checked whether it dominates the existing label at the current position (Lines 10-12). If not, the label is added into the existing list at the current position (Lines 13-16). If the potential label does dominate the existing label, the existing label is removed and the condition is checked for the following label in the next position, and this is repeated until a nondominated existing label is encountered. At this point, the potential label is added to the list at the current position (Lines 13-16). After a potential label is added, the position is incremented because the next potential label is mutually nondominating with the label that was just added.

The ConstructPaths function is described in Algorithm 3.7. ConstructPaths takes each label at an order node or a late node satisfying an order and constructs the path that

Algorithm 3.6 Nondominated Labels Function

procedure ADDNONDOMLABELS(n, j, pls)
    $pos \leftarrow 0$ ▷ Initialize list position of ex. label
    for all $pl \in pls$ **do** ▷ For each label in pot. labels list
        while $pos < length(NI(j)) \wedge rc(pl) > rc(IL(NI(j)(pos)))$ **do** ▷ While not at end and pot. rc exceeds ex. rc
            $pos \leftarrow pos + 1$ ▷ Shift to next ex.
        end while
        if $rc(pl) = rc(IL(NI(j)(pos))) \wedge lr(pl) < lr(IL(NI(j)(pos))) \vee pos > 0 \wedge lr(pl) < lr(IL(NI(j)(pos - 1)))$ **then** ▷ If an ex. label dominates the pot. label
            ContinueFor ▷ Do not add pot., continue to next pot.
        end if
        while $pos < length(NI(j)) \wedge rc(pl) < rc(IL(NI(j)(pos))) \wedge lr(pl) \geq lr(IL(NI(j)(pos))) \vee rc(pl) \leq rc(IL(NI(j)(pos))) \wedge lr(pl) > lr(IL(NI(j)(pos)))$ **do** ▷ While pot. dominates ex. at current pos
            DELETE$(IL(NI(j)(pos)))$, DELETE$(NI(j)(pos))$ ▷ Remove dominated ex.
        end while
        $IL(n) \leftarrow pl$ ▷ Instantiate pot.
        INSERT$(NI(j), pos, n)$ ▷ Add pot. at current position in labels
        $n \leftarrow n + 1$ ▷ Increment current label index
        $pos \leftarrow pos + 1$ ▷ Shift to next position in labels
    end for
    return n
end procedure

corresponds to the label. The path that is constructed is known to be a feasible and negative reduced cost path that is Pareto-optimal in terms of reduced cost and shelf-life remaining, because only labels satisfying these conditions are maintained in the lists at each node. The path is constructed backwards from the label in an iterative fashion by taking the predecessor index of a current label and hashing into the IL dictionary to retrieve the predecessor label. To calculate the cost parameter for the path that is needed in the second stage commodity routing formulation, the flow costs of edges in the path are added cumulatively as the path is constructed. Once constructed, the path and its flow cost parameter are added as a pair to the set of new paths $\mathcal{P}_k^{new}$. CONSTRUCTPATHS returns this set of new paths to FINDPATHS, which immediately returns it to the cutting planes algorithm.

Algorithm 3.7 Path Construction Function

procedure ConstructPaths($IL, NI, \mathcal{P}_k$)
    $\mathcal{P}_k^{new} = \emptyset$ ▷ Begin with empty new paths set
    for all $(w,t,g) \in \mathcal{O}_k$ **do** ▷ For each order
        for $t^{late} \in \{0,1,\ldots,t^{late}_{(w,t,g)}\}$ **do** ▷ For each time step satifying order
            for $n' \in NI(w, t+t^{late})$ **do** ▷ For each index at (late) order node
                if $o(n') = (w,t,g)$ **then** ▷ If order of label matches current order
                    $j \leftarrow (w, t+t^{late})$ ▷ Initialize current node variable
                    $ppairs \leftarrow list(j, status(IL(n')))$ ▷ Initialize path (node, status) list
                    $cost \leftarrow t^{late} \times c^{late}_{(w,t,g)}$ ▷ Begin with cost equal to lateness penalty
Build path pairs list from (late) order node backwards to good-supplier node
                    while $n' \neq 0$ **do** ▷ Loop until reach label at good-supplier node
                        $i \leftarrow node(IL(n'))$ ▷ Update next node variable
                        $n' \leftarrow pred(IL(n'))$ ▷ Update current index to predecessor index
                        $cost \leftarrow cost + c_{(i,j)}$ ▷ Update cumulative cost with edge cost
                        $j \leftarrow node(IL(n'))$ ▷ Get node corresponding to label index
                        Append($ppairs, (j, status(IL(n')))$) ▷ Add new pair to list
                    end while
                    $(\_, s, t') \leftarrow i$ ▷ Get supplier and time for good-supplier node i
                    $cost \leftarrow cost + c_{((s,t'),(g,s,t'))}$ ▷ Update cost with purchase cost
                    Append($ppairs, ((s,t'), False)$) ▷ Add pair for supplier node to list
                    Append($ppairs, (`src', False)$) ▷ Add pair for supersource to list
                    Reverse($ppairs$) ▷ Reverse list so ordered from supersource
                    if $(ppairs, cost) \notin \mathcal{P}_k$ **then** ▷ If path not in restricted set of paths
                        $\mathcal{P}_k^{new} \leftarrow \mathcal{P}_k^{new} \cup (ppairs, cost)$ ▷ Add path to new paths set
                    end if
                end if
            end for
        end for
    end for
    return $\mathcal{P}_k^{new}$ ▷ Return unique new paths
end procedure

4 Computational Testing

4.1 Software and Solvers

The Benders decomposition, constraint generation, and multilabel constrained shortest path algorithms were coded in Python 2.7. NetworkX is a Python package "for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks", and it was used to represent the graph G. Gurobi 5.0.1 was used to solve the MIP *1R-R* and the continuous LPs *2CR-DR(k)* on a quad-core Intel i7 processor with 8GB of memory.

The basic parameters for warehouses, suppliers, and edges are inputted or created in a GIS (geographical information systems) software package called Quantum GIS. The data is in GIS table format and is exported to CSV files. Quantum GIS produces geographical visualizations of the supply chain such as Figure 12.

4.2 Network Graph Generation

In Quantum GIS, warehouses and suppliers are placed on a world map and parameters such as supply and inventory costs, capacities, and inventory storage shelf-life expenditures are inputted to columns in GIS database tables. Nodes are then connected via multisegment arcs corresponding to transport routes, and using GIS functions, the lengths of the multisegment arcs are calculated and saved in the arc database table. This is not the distance between the nodes, but the sum of the length of the segments of an arc, so it is representative of the distances traveled by ships or land vehicles on roads that are not 'as the crow flies'.

The GIS data tables are exported to CSV format and read into spreadsheets. In spreadsheet format, the node and transport and arc route data is manipulated with simple functions used to estimate costs of transport, the shelf-life expenditure on transport routes and during refrigerated and nonrefrigerated storage at warehouses, and other node and edge parameters needed to construct G. In a separate file, the number of time steps $|\mathcal{T}|$ in the time horizon of the decision-maker is specified. For the 'base class' used in computational testing, 108 time periods are used, representing 1 week time steps over the course of a two-year period plus an initialization period of four weeks. A Python function is called to automate the process of constructing G from the GIS data.

The World Food Programme does not have a public database with information about its warehouses, suppliers, and transport routes. There is little data available on demand and need for SNPs since they are relatively new and production falls short of need. Information gathered from conversations with WFP employees and from the WFP website has been combined with guesswork to create data for several international suppliers, country- and regional-level warehouses, and directed transport routes in a GIS database. Figure 12, produced in Quantum GIS, represents this information on a map of the world. Information from the GIS databases is excerpted in Tables 10 to 12.

Table 10: Suppliers Data

$s \in \mathcal{S}$	m	$(c, m)_{g7}$	$(c, m)_{g13}$	$(c, m)_{g26}$	$(c, m)_{g52}$
sFra	7000	(20, 5000)	(25, 7000)	(30, 3000)	(0, 0)
sPak	2000	(0, 0)	(0, 0)	(25, 1000)	(20, 2000)
sGha	800	(15, 600)	(20, 800)	(0, 0)	(35, 200)
sBra	3000	(0, 0)	(30, 3000)	(40, 2000)	(0, 0)
sTan	2000	(20, 1500)	(0, 0)	(10, 2000)	(0, 0)

Table 11: Warehouses Data

$w \in \mathcal{W}$	c^r_w	c_w	m_w	l^r_w	l_w
pAlg	5000	0.13	8000	0.2	0.8
pAng	6000	0.33	4000	0.2	1
pBan	9000	0.23	6000	0.2	1.1
pBra	8000	0.33	7000	0.2	0.8
pChi	4000	0.2	6000	0.2	0.9
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 12: Transport Routes Data

$i \in \mathcal{N}$	$j \in \mathcal{N}$	$t' - t$	$c_{(i,j)}$	$m_{(i,j)}$	$l_{(i,j)}$
pAlg	wAlg	2	2	750	3
pAng	wZam	1	1.6	250	1.5
pAng	wNam	1	1.6	250	1.5
pBan	wIndi	1	1.6	1500	1.5
pBan	wBhu	1	1.2	500	1.5
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

4.3 Stochastic Scenario and Order Generation

Order generation requires a set of parameters on the good types, and these are specified in a spreadsheet file. Goods are likely to be needed with different frequencies, and it is assumed that these relative frequencies are consistent over scenarios, so a probability of an order having good type $g \in \mathcal{G}$ is specified. Each good type g has two parameters for the lower and upper bounds on the maximum shelf-life expenditure on paths to a node (w, t) corresponding to order (w, t, g). Finally, each good type g has two parameters for the lower and upper bounds on the number of late periods for which demand for the good can be satisfied. The goods differ in their shelf-lives, and scientific studies are still being conducted to ascertain how the nutritional qualities of specialized nutrition products depend on storage conditions. For the base class, Table 13 displays the goods data. Four good types are used with four different shelf-lives, and it is assumed the shelf-lives are for average tropical atmospheric conditions. Goods are named according to their shelf-lives, so at the lower extreme, $g7$ has 7 weeks of shelf-life and at the upper extreme, $g52$ has 1 year of shelf-life.

Table 13: Goods Data

$g \in \mathcal{G}$	$pr(good)_g$	max $l_{(w,t,g)}$	min $l_{(w,t,g)}$	min $t^{late}_{(w,t,g)}$	max $t^{late}_{(w,t,g)}$
$g7$	0.2	5	2	1	2
$g13$	0.3	8	2	1	3
$g26$	0.3	16	4	2	3
$g52$	0.2	37	4	2	5

Edge flow cost, capacity, and shelf-life expenditure parameters are assumed to be constant over time, and orders and demand quantities are stochastic. The number of scenarios to be generated, $|\mathcal{K}|$, is varied to determine how the algorithms scale, and for the base class four scenarios are used. Each scenario $k \in \mathcal{K}$ has orders drawn from the same uniform distribution over time-expanded warehouses. Associated with k is the scenario probability α_k, randomly chosen and normalized such that $\sum_{k \in \mathcal{K}} \alpha_k = 1$.

An important parameter $pr(node)$ describes the probability that a time-expanded warehouse node (w, t) is selected as a potential order node. This parameter determines the expected number of orders on the network over the time horizon, and it is varied in the next section to determine how the algorithms scale with number of orders. For the base class, $pr(node) = 0.01$. No orders are generated on time-expanded warehouse nodes during the 4 weeks of initialization. A parameter $pr(order) = 0.5$ specifies the probability of each additional good $g \in \mathcal{G}$ being selected to potentially create an order (w, t, g) at the node (w, t). If the good g is selected for the node, then the probability that an order (w, t, g) is actually created at the node is determined by $pr(good)_g$, which differs for each good and is specified in Table reftab:goods. There can only be one order per good type at a node, but there can be multiple orders at a node, each corresponding to a different good type.

Once an order (w, t, g) is created, the order parameters need to be generated from distributions. The demand quantity d_o associated with order $o = (w, t, g)$ is generated from a truncated normal distribution defined by a minimum demand bound $min(d) = 500$ units, a mean demand $mean(d) = 3000$ units, and a standard deviation of demand $sd(d) = 2000$ units. The cost of unmet demand c^u_o is generated from a uniform distribution between lower and upper bound parameter values, $min(c^u) = 10$ currency units and $max(c^u) = 100$ currency units.

The directed acyclic graph G for the base class has $|\mathcal{N}| = 10876$ nodes, $|\mathcal{E}| = 31771$ edges, and an average in-degree equal to average out-degree of 2.92. The number of warehouses is 75, and since there are 108 periods, there are 8100 time-expanded warehouse nodes. The base class instance has the following numbers of scenarios: $|\mathcal{O}|_1 = 45, |\mathcal{O}|_2 = 38, |\mathcal{O}|_3 = 45, |\mathcal{O}|_4 = 55$. Using this problem instance, several different options for parameters governing the Benders decomposition and second stage cutting planes algorithms are tested. The standard options used for algorithmic testing are default automatic perturbation, no manual perturbation, no removing of path cuts in the second stage problems, and solving a single second stage problem after each first stage iteration (cycling over the scenarios). The optimality gap metric used in testing was the average over each scenario $k \in \mathcal{K}$ of the relative gap between the second stage optimal objective Z'_k and the first stage maximum cost variable Z_k:

$$\text{average relative gap} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \frac{(Z'_k - Z^*_k)}{Z'_k}$$

4.4 Options for Removing Paths in the Second Stage Problem

Several different rules for removing path cuts from the relaxed second stage dual problems were tested. Cuts could be removed either once immediately after a Benders iteration ('initial drop rule'), or between every iteration of the second stage cutting planes algorithm ('looping drop rule'), or both. Solve time was averaged over five runs of the algorithm. The rules tested were as follows.

ds delete paths with slack higher than 50.0 (from experience with the base class, this value on average removed the 10% of cuts with the highest slack)

ka keep all path cuts from previous solve

Table 14: Effect of Rules for Removing Path Cuts Before Solving Second Stage Problem

initial drop rule	looping drop rule	average relative gap	time (s)
ka	ka	0.000	282
ka	ds	0.000	275
ds	ka	0.000	332
ds	ds	0.000	318

For each combination of rules used, the problem solved to optimality. It appears that computational performance for the base class was improved by keeping all cuts from the last solve following a Benders master iteration. Deleting paths with a high slack between every second stage cutting planes iteration seems to have had little effect.

4.5 Benders Subproblems to Solve after a Master Iteration

Three different rules for selecting which of the $|\mathcal{K}|$ second stage problems *2CR-DR(K)* to re-solve after each Benders master iteration are tested. Note that any of the selected *2CR-DR(K)*s must currently satisfy the second stage problem re-solve condition, that the objective value Z'_k of the *2CR-DR(K)* exceeds the corresponding variable Z_k in the *1R-R*.

Rule 'all' solves all of the second stage problems satisfying the re-solve condition before another *1R-R* iteration. This means that the same set of refrigeration decisions ($\bar{\mathbf{R}}$) is given to all the *2CR-DR(K)*s. The 'next' rule picks the next $k \in \mathcal{K}$ that is in the set of *2CR-DR(K)*s satisfying the re-solve condition. This rule cycles over the *2CR-DR(K)*s that need to be re-solved, regardless of the size of the relative gap, so it takes a breadth-first approach. Finally, the 'largest gap' rule is similar to the 'next' rule in that it only solves one *2CR-DR(K)* at a time, but the way it selects the k is different. It picks the subproblem for k such that the relative gap $\frac{(Z'_k - Z^*_k)}{Z'_k}$ between the *2CR-DR(K)* objective value and optimal value of the corresponding *1R-R* variable Z^*_k is largest. This rule takes a more depth-first approach than the 'next' rule. Note that parallelization of the solution process of multiple *2CR-DR(K)*s is possible when using the 'all' rule, but this was not implemented.

Table 15: Effect of Different Second Stage Problem Re-solve Rules

rule	average relative gap	time (s)	iters for *1R-R*	iters for *2CR-DR(k)*
all	0.000	370	24	220, 93, 162, 303
next	0.000	281	38	180, 74, 98, 286
largest gap	0.000	308	37	185, 82, 173, 279

The 'next' rule appears to be the most effective in terms of overall solve time for the base class. The 'all' rule required fewer Benders master iterations overall but a greater number of second stage iterations. This is because multiple cuts are added between each master iteration, so more information is received by the Benders master problem from the second stage problems, but the second stage problems receive updated refrigeration decisions less frequently and thus require a greater number of solves. Overall, the 'all' rule performed significantly worse than the 'next' rule. The 'largest gap' rule was slightly slower than the 'next' rule, taking a similar number of master iterations but requiring significantly more solves for only one of the scenario subproblems.

4.6 Testing Scaleability of Algorithms with Different Problem Classes

Through testing different algorithmic options on the base class, a relatively efficient set of algorithmic options was found. For testing on different problem classes, these options are kept constant. The 'next' rule is used for selecting the set of second stage problems to re-solve after each Benders master problem, the 'keep all' rule is used to drop path cuts once after each Benders iteration, and the 'delete high slack' rule is used between every iteration of the cutting planes algorithm.

In this section, the size of the *RCR* problem is varied along several important dimensions to determine how the solution approach scales with problem size. This is useful for gaining insight into the bottlenecks of the algorithm and what types of problems it is likely to be capable of solving efficiently. Problem size is varied according to the number of scenarios ($|\mathcal{K}|$), the number of post-initialization periods ($|\mathcal{T}|$), and the expected number of orders per scenario ($\mathbb{E}(|\mathcal{O}|)$, through changing $pr(node)$). Along each dimension, three different values parameter values ('low', 'med', and 'high') are tested. In each case, the 'med' level

Table 16: Problem Classes

	class	scenarios ($\|\mathcal{K}\|$)	periods ($\|\mathcal{T}\|$)	$pr(node)$
	base	4	108	0.01
Vary $\|\mathcal{K}\|$	k-low	**1**	108	0.01
	k-med	**4**	108	0.01
	k-high	**16**	108	0.01
Vary $\|\mathcal{T}\|$	t-1y	4	**56**	0.01
	t-2y	4	**108**	0.01
	t-4y	4	**212**	0.01
Vary $pr(node)$: changes $\mathbb{E}(\|\mathcal{O}\|)$	o-low	4	108	**0.0025**
	o-med	4	108	**0.01**
	o-high	4	108	**0.04**

correponds to the base class, the 'low' level to half of base class parameter, and the 'high' level to double the base class parameter. Note that the number of periods parameter values have the initialization periods (4 weeks in each case) added after the base class parameter of 104 non-initialization periods is doubled and halved to get the 'high' and 'low' levels respectively.

Table 17: Solution Statistics for Different Numbers of Scenarios

class	$\|\mathcal{K}\|$	average relative gap	time (s)	iters for *1R-R*
k-low	**1**	0.000	92	21
k-med	**4**	0.000	281	38
k-high	**16**	0.000	833	90

Table 18: Solution Statistics for Different Time Horizons

class	$\|\mathcal{T}\|$	average relative gap	time (s)	iters for *1R-R*
t-1y	**56**	0.000	36	29
t-2y	**108**	0.000	281	38
*t-4y	**212**	0.250	-	228

Solve terminated because time exceeded 45 minutes

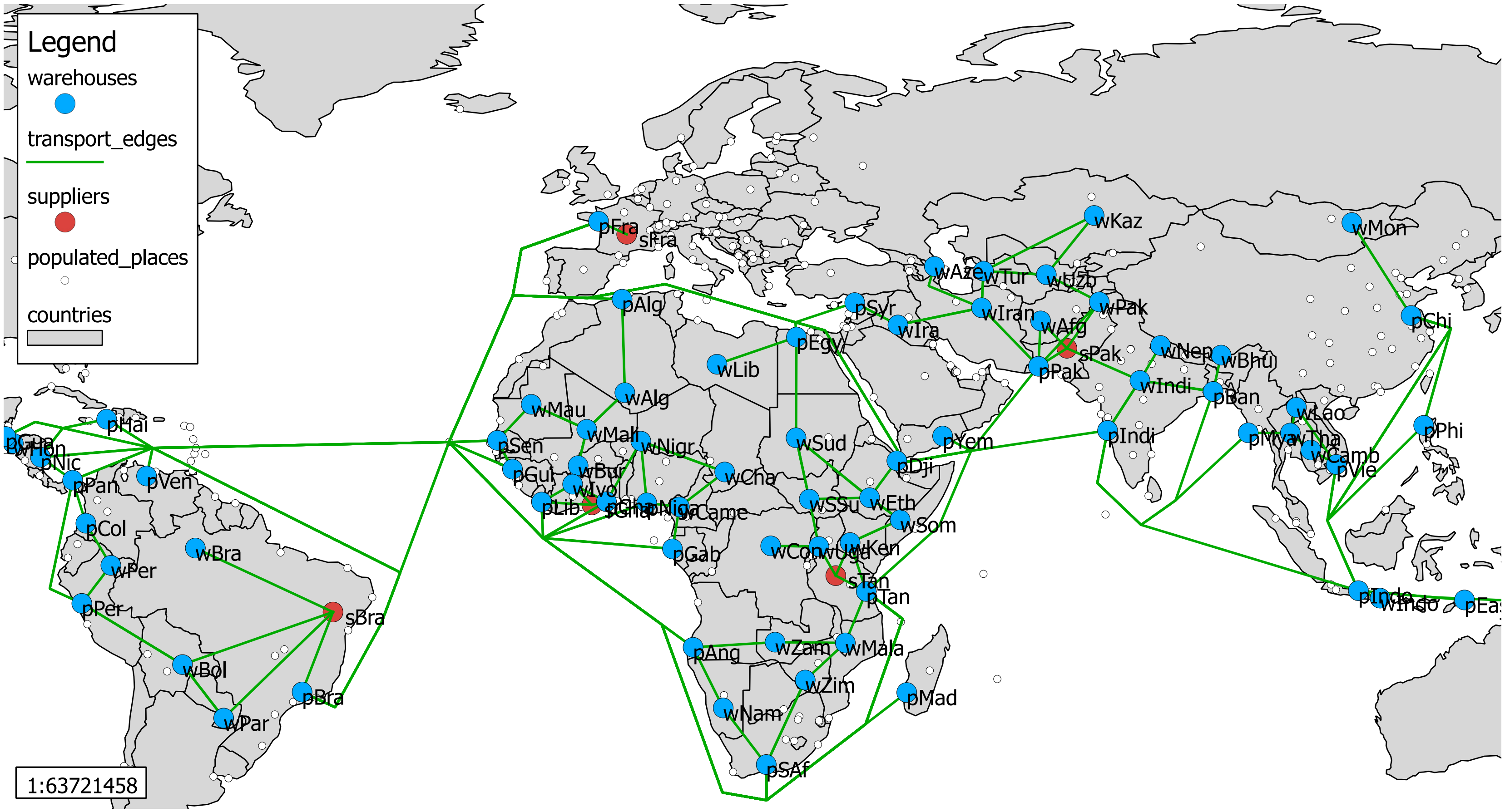
The computational results in Tables 17 to 19 suggest that solution time increases slightly slower than linearly with the number of scenarios and with the expected number of orders. The number of Benders iterations also increased approximately linearly with the number of scenarios, but was not significantly affected by the expected number of orders. Doubling the length of the time horizon caused what appears to be an exponential increase in problem difficulty as measured by both solution time and the number of Benders master iterations. While a problem on a network with 56 time periods was solved in a little over half a minute,

Table 19: Solution Statistics for Different Expected Numbers of Orders

class	$pr(node)$	average relative gap	time (s)	iters for *1R-R*
o-low	**0.0025**	0.000	83	31
o-med	**0.01**	0.000	281	38
o-high	**0.04**	0.000	949	37

a problem with double the number of time periods took almost five minutes, and a further doubling caused the problem to take longer than 45 minutes to solve. These results make sense because the number of paths is exponential in the size of the graph G, and the only dimension tested that affects the size of G was the number of time periods.

Figure 12: Map of the Suppliers, Warehouses, and Transport Routes used to Construct G



5 Conclusions and Future Work

The World Food Programme, in considering how to expand the distribution of specialized nutrition products, faces the problem of deciding which warehouses to refrigerate and how to route these perishable goods through its distribution network. A large-scale two-stage stochastic program was formulated for the refrigeration and commodity routing problem (RCR) on a time-expanded network. This model is an example of the fixed-charge network problem, which is known to be NP-hard.

A Benders decomposition was used to separate the first stage refrigeration problem and the second stage commodity routing problem for each scenario. Following a re-definition of path variables, a cutting planes algorithm was used to solve the second stage problems using a novel multilabel constrained shortest path algorithm. The overall solution procedure was shown to be powerful, leading to an optimal solution on a realistic problem size of 75 warehouses, 108 time periods, and nearly 200 demand orders in four scenarios in around five minutes.

Future work will involve developing a decision tool for the World Food Programme's operational and tactical level decision-making on the distribution network for specialized nutrition products. A multi-period stochastic model would better account for the uncertainty that the WFP faces. A decision problem that makes the pre-positioning of goods at warehouses in preparation for natural disasters would benefit from such a multi-period model of uncertainty. Since hunger in some parts of the world is highly seasonal, demand for goods could be modeled on annual cycles.

A deeper theoretical and practical analysis of the complexity of the multilabel constrained shortest path algorithm developed for this problem is called for. Parallelization of the algorithm, the use of more efficient data structures, and implementing stronger bounding ideas for paths would likely lead to improvements in the solve time.

References

[1] A. Afshar and A. Haghani. Modeling integrated supply chain logistics in real-time large-scale disaster relief operations. *Socio-Economic Planning Sciences*, 2012.

[2] R. Agarwal and . Ergun. Ship scheduling and network design for cargo routing in liner shipping. *Transportation Science*, 42.2:175–196., 2008.

[3] R. Alvarenga, Ö. Ergun, J. Li, F. Mata, N. Shekhani, D. Slaton, J. Stone, A. Vasudevan, and E. Yang. World food programme east african corridor optimization, senior design final report, georgia institute of technology, h. milton stewart school of industrial and systems engineering, atlanta, ga. 2010.

[4] V. De Angelis, M. Mecoli, C. Nikoi, and G. Storchi. Multiperiod integrated routing and scheduling of world food programme cargo planes in angola. *Computers & Operations Research*, 34(6):16011615, 2007.

[5] A. Armacost, C. Barnhart, and K. Ware. Composite variable formulations for express shipment service network design. *Transportation Science*, 36:1:1–20, 2002.

[6] C. Barnhart, C. Hane, E. Johnson, and G. Sigismondi. An alternative formulation and solution strategy for multi-commodity network flow problems. *Telecommunications Systems*, 3:239258, 1995.

[7] C. Barnhart, N. Boland, L. Clarke, E. Johnson, G. Nemhauser, and R. Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32:208220, 1998.

[8] C. Barnhart, A. Cohn, E. Johnson, D. Klabjan, G. Nemhauser, and P. Vance. Airline crew scheduling. *Handbook of Transportation Science*, pages 517–560, 2003.

[9] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252., 1962.

[10] D. Berkoune, J. Renaud, M. Rekik, and A. Ruiz. Transportation in disaster response operations. *Socio-Economic Planning Sciences*, 46:2332, 2012.

[11] S. Binato, M. Pereira, and S. Granville. A new benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems*, 16:235–240, 2001.

[12] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34:1, January 2006:5868, 2006.

[13] M. Celik, O. Ergun, B. Johnson, P. Keskinocak, A. Lorca, P. Pekgun, and J. Swann. Humanitarian logistics. In *INFORMS, Phoenix 2012*, 2012.

[14] UN Logistics Cluster. About the logistics cluster. URL `http://www.logcluster.org/about/logistics-cluster/`.

[15] J. Cordeau, F. Soumis, and J. Desrosiers. Simultaneous assignment of locomotives and cars to passenger trains. *Operations Research*, 49:531548, 2001.

[16] J. Cordeau, G. Laporte, M. Savelsbergh, and D. Vigo. *Handbook in Operations Research and Management Science: Transportation, volume 14*, chapter Vehicle Routing. Elsevier, 2006.

[17] J. Cordeau, F. Pasin, and M. Solomon. An integrated model for logistics network design. *Annals of Operations Research*, 144: 1:59–82, 2006.

[18] Alysson M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers and Operations Research*, 32(6):1429–1450, June 2005.

[19] T. Crainic. Service network design in freight transportation. *Eur. J. Oper. Res*, 122: 272288, 2000.

[20] M. Desrochers. An algorithm for the shortest path problem with resource constraints, technical report les cahiers du gerad. Technical report, University of Montreal, Montreal, 1988.

[21] M. Desrochers and F. Soumis. A generalized permanent labeling algorithm for the shortest path problem with time windows. *INFOR*, 26:191212, 1988.

[22] F.G. Engineer, G.L. Nemhauser, and M.W.P. Savelsbergh. Shortest path based column generation on large networks with many resource constraints. Technical report, Technical report, Georgia Tech, College of Engineering, School of Industrial and Systems Engineering, 2008.

[23] Food and Agriculture Organization of the United Nations International Fund for Agricultural Development World Food Programme. The state of food insecurity in the world 2012, 2012. URL `http://www.fao.org/publications/sofi/en/`.

[24] M. Garey and D. Johnson. *Computer and intractability: A guide to the theory of NP completeness*. Freeman, San Francisco, 1979.

[25] B. Gendron, T. Crainic, and A. Frangioni. *Multicommodity capacitated network design*, chapter Telecommunications Network Planning, pages 1–19. Kluwer, 1999.

[26] A Geoffrion and G. Graves. Multicommodity distribution system design by benders decomposition. *Management Science*, 20:822–844, 1974.

[27] P. Van Hentenryck, R. Bent, and C. Coffrin. *Strategic planning for disaster recovery with stochastic last mile distribution. In Lecture Notes in Computer Science: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer-Verlag, Berlin, 2010.

[28] K. Holmberg and D Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *Informs Journal on Computing*, 15: 4257, 2003.

[29] Earth Policy Institute. By the numbers data highlights from full planet, empty plates, October 2012. URL http://www.earth-policy.org/data_highlights/2012/highlights32. Accessed Nov 16, 2012.

[30] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. *Column Generation*, pages 33–65, 2005.

[31] F. Liberatore, M.T. Ortuno, G. Tirado, B. Vitoriano, and M. P. Scaparra. A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics. *Computers & Operations Research*, 2012.

[32] from Wikimedia Commons Lobizn at en.wikipedia. Percentage population undernourished world map, May 2008. URL http://upload.wikimedia.org/wikipedia/commons/7/78/Percentage_population_undernourished_world_map.PNG.

[33] T. Magnanti and R. Wong. Network design and transporation planning: Models and algorithms. *Transportation Science*, 18:155, 1984.

[34] Nutriset. Cronobacter sakazakii, the reaction of nutriset, . URL http://www.nutriset.fr/en/production/cronobacter-sakazakii-la-r%C3%A9action-de-nutriset.html. Accessed Nov 16, 2012.

[35] Nutriset. Plumpynut and the cmam approach, . URL http://www.nutriset.fr/en/innovation/plumpynut-cmam-example.html.

[36] Ravindra Ahuja Thomas Magnanti James Orlin. *Network Flows*. Prentice Hall, 1993.

[37] UN World Food Programme. Un humanitarian response depot, . URL http://www.hrdlab.eu/.

[38] UN World Food Programme. What is malnutrition?, . URL http://www.wfp.org/hunger/malnutrition. Accessed Nov 16, 2012.

[39] World Food Programme. About - faqs, . URL http://www.wfp.org/faqs.

[40] World Food Programme. Food specifications, . URL http://foodquality.wfp.org/FoodSpecifications/tabid/56/Default.aspx.

[41] World Food Programme. Wfp nutrition policy, nutrition programs and food supplements, . URL http://documents.wfp.org/stellent/groups/public/documents/resources/wfp247204.pdf. Accessed Nov 16, 2012.

[42] World Food Programme. Special nutritional products, . URL http://www.wfp.org/nutrition/special-nutritional-products. Accessed Nov 16, 2012.

[43] World Food Programme. Food procurement annual report 2011. http://documents.wfp.org/stellent/groups/public/documents/communications/wfp244715.pdf, 2011.

[44] World Food Programme. Guidance for temperature control storage & transportation (nutritious food products). March 2012.

[45] L. Consuelos Salas, M. R. Cardenas, and M. Zhang. Inventory policies for humanitarian aid during hurricanes. *Socio-Economic Planning Sciences*, 2012.

[46] R. Sandhu and D. Klabjan. Integrated airline fleeting and crew-pairing decisions. *Operations Research*, 55:439456, 2007.

[47] A. Thomas and M. Mizushima. Logistics training: necessity or luxury? *Forced Migration Review*, 22:6061, 2011.

[48] P. Tsamasphyrou, A. Renaud, and P. Carpentier. Transmission network planning under uncertainty with benders decomposition. *Lecture Notes in Economics and Mathematical Systems*, 481:457–472, 2000.

[49] Dimitris Bertsimas John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[50] Emma Rowley Garry White. World on track for record food prices 'within a year' due to united states drought, September 2012. URL `http://www.telegraph.co.uk/finance/commodities/9561143/World-on-track-for-record-food-prices-within-a-year-due-to-US-drought.html`. Accessed Nov 16, 2012.