Intra Market Optimization for Express Package Carriers with Station to Station Travel and Proportional Sorting

Luke Schenk (*lukeschenk@gmail.com*)
Cap Geminini U.S.
Chicago, IL

Diego Klabjan (*d-klabjan@northwestern.edu*)
Department of Industrial Engineering and Management Sciences
Northwestern University, Evanston, IL

The flow of packages of an express package carrier consists of pick ups at costumer locations by couriers and delivering the packages to a local station for sorting. The packages are then transported to a major regional sorting facility called the ramp. At the ramp, packages can be sorted again before departing to a hub. From the hub they are moved to the destination ramp, where the entire process repeats in the reverse order until ultimate delivery of the package to the end customer. We focus on the afternoon and evening operations concerning stations and the ramp. Sorting and transportation decisions among these locations are considered. The most important decisions are: (1) which packages to aggregate at the stations, and (2) what is the most efficient transportation among locations to meet time deadlines at the ramp. Several options for modeling the sorting process at stations and the ramp, as well as the possibility of vehicles traveling from one station to another station to consolidate volume before proceeding to the ramp are considered. We model these processes by means of a dynamic program, where time periods represent time slices in the afternoon and evening. The overall model is solved by approximate dynamic programming, where the value function is approximated by a linear function. Further strategies are developed to speed up the algorithm and decrease the time needed to find feasible solutions. The methodology is tested on several instances from an express package carrier. The dynamic program solutions are substantially better than the current best practice and the best solutions obtained from an integer programming formulation of the problem.

1. Introduction

Packages move through several steps within the originating market before being loaded on a truck or plane to move through one or more hubs, after which the packages arrive at the destination market. Those tasks that occur within the market where the package originated are known as *intra market operations*. It is important to route and sort packages within the origin market in such a manner that they meet their departure times to prevent flight delays from propagating throughout the network. Customers use express shipment to ensure on-time delivery, so the failure of packages to depart their origin market can lead to poor customer service levels. While it is critical to deliver packages in a timely manner, excessive handling and package routing can lead

to high operating costs. Figure 1 provides a depiction of intra market operations with the possibility of aggregating volume at an intermediate station before proceeding to the ramp.

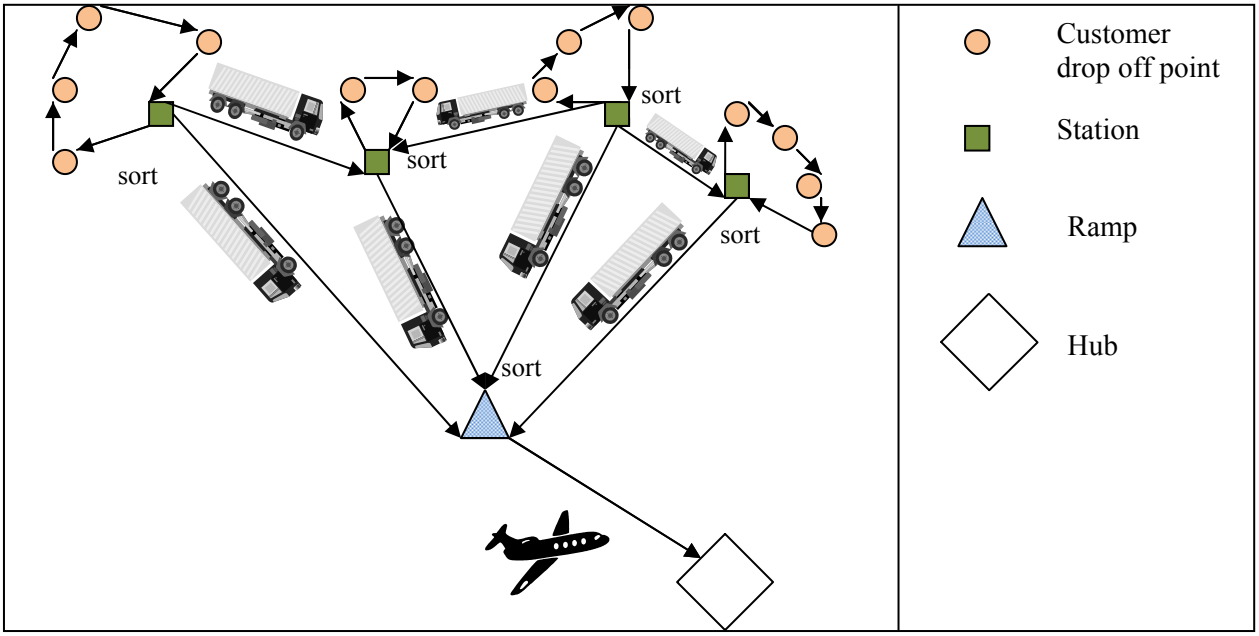

Figure 1: Intra Market Depiction

The intra market operations for moving packages outbound from the market occur in the afternoon and evening hours to meet overnight departure times. Therefore the problem of moving the volume from stations to the ramp is referred to as the *PM operations problem*. This problem consists of making the sorting decisions, determining how to create containers, and transportation decisions such as which transportation mode to use and which containers to load. A *multi-class package flow* refers to a set of packages going to the same outbound destination and of the same package type (box, letter). The term multi-class refers to the fact that they are differentiated by package type and the term flow stands for the fact that it is a set of packages with a specific origin and destination, and the underlying routing. For example, a set of letters going from Peoria, IL to Fresno, CA represents a single multi-class package flow. If a container is created at a station consisting of a single multi-class package flow, then such a *pure container* need not be sorted again at the ramp and therefore cost savings occur. On the other hand, such decisions can yield higher transportation costs due to the use of more containers and thus increased transportation capacity.

Within a season, the daily inflow of packages at stations is fairly stationary and it does not fluctuate from day to day. As a result, the planning problem under consideration is solved before the beginning of a season and it is typically not altered throughout the season. During a season, based on the provided solution of the proposed model, the macro level decisions are deployed:

- every morning, the recommended vehicles are dispatched to stations,
- each vehicle is preloaded with the assigned number of containers, and
- during sorting at stations, the correct mix of pure vs. mixed containers is attempted.

Another reason for adhering to the same plan throughout the season is operational ease and efficiency. Clearly such high repetition is possible due to stationary inflow of packages.

In Schenk and Klabjan (2008), the authors formulate a version of the PM operations problem as a dynamic program and propose an algorithm to solve the problem based on approximating the value function by a linear function. In their model, the sorting process assumes that whenever a sorting queue builds up, it is possible to select the volume to be processed next in an optimal way (prioritized sorting). This is only an approximation since in real operations most of the time the sorting queue follows the first-in first-out principle. In this manuscript we introduce a new concept termed *proportional sorting*, which models the sorting process more accurately. Proportional sorting introduces nonlinearities, which complicates the solution methodology. Schenk and Klabjan (2008) also assume that a transportation vehicle travels directly from a station to the ramp. In this work we relax this restriction by allowing a vehicle to start at a station and then travel to a different station to pick up additional volume or to unload volume before proceeding to the ramp. This substantially complicates the model, but, can lead to further cost savings through consolidation and a reduced number of vehicles to transport the packages to the ramp.

This paper makes the following contributions.

1. A detailed model of the PM operations problem that includes the option of aggregating volume between stations and more detailed sorting and unloading options.
2. An approximate dynamic programming strategy for solving the model.
3. Experimental work that demonstrates the quality of the solution obtained by the algorithm.

Results of computational experiments show that linear approximations using gradient information generate good solutions to problems with many more time periods, longer travel times, and larger, more heavily constrained state and action spaces than other previously studied problems where this method was implemented. Since optimal solutions to the PM operations problem even for smaller markets are not known, solutions are benchmarked against the current operations in practice. Strategies are presented for choosing initial values for the value function approximation and other methods to improve the speed and convergence of the algorithm.

Section 2 gives a review of existing literature on dynamic programming and transportation problems, while Section 3 discusses the PM operations problem in greater detail, including small examples showing alternative sorting options and the benefits of station to station travel. Section 4 outlines the states, actions, and costs that are captured in the dynamic programming formulation of the PM operations problem. While some example system dynamics and action space constraints are given, the dynamic program is very complex and the reader is referred to Schenk (2005) for the complete mathematical model. Section 5 gives the general overview of the approximate dynamic programming algorithm and gives details as to how the corresponding solution methodology is modified to include the extensions to the model. Section 6 presents the results of computational experiments for the dynamic programming algorithm with station to station travel. These results are compared to the integer-programming-based algorithm found in Schenk and Klabjan (2008) as well as current operations in practice.

2. Literature Review

Our working is related to three areas: freight transportation, express package operations and planning, and approximate dynamic programming. There is abundant literature on designing service networks in freight transportation, see, e.g., surveys Crainic (2003), Crainic and Kim (2007). Our work is not really about designing the network, but, the common theme is the selection of a mode of transportation. Another important difference is the granularity and the time ho-

rizon. Service network design problems deal with a multi-year time horizon, while our discretization is down to minutes of a single afternoon. The employed modeling and algorithmic techniques are vastly different and are much more aligned with the work by Powell et al. (2007), which is discussed in greater details later.

There is limited literature concerning express package deliveries. Kim et al. (1999) model multimodal express package delivery through a model which they call the Express Shipment Service Network Design Problem (ESSND). This paper presents solution methodologies for moving packages from their origins through their originating market to their destinations with time windows for delivery. This work routes packages from the ramp to the hub while considering package transfer at an intermediate destination, and can similarly be used for routing from the hub to the destination ramp on the outbound side. The volume of packages is considered the same from day to day in order to allow for flow conservation of aircraft. The problem modeled in our paper can be thought of as the input to the ESSND problem. Several problem reduction methods and heuristics are presented to decrease problem size without compromising model optimality. Further solution methodologies to the ESSND problem are given in Armacost et al. (2002). This paper introduces the notion of composite variables, which are essentially variables that capture both aircraft routing and package flow decisions. There is thus no longer a need for separate variables representing these decisions and the problem size and solution time are reduced considerably. Both papers conclude that integer programming methods must be combined with heuristic strategies and other problem modifications to solve large problem instances.

Our paper draws on the emerging field of approximate dynamic programming. For problems modeled as a dynamic program with many decisions and a large state space, the number of possible outcomes grows considerably and makes problems difficult to solve. These problems such as the PM operations problem with a large state space require the use of approximate dynamic programming to provide an estimate to the value function. Much of the recent work on dynamic programming applications and approximation algorithms to transportation problems is summarized in Powell et al. (2007) and Powell and Topaloglu (2003). Powell et al. (2001) provides a formal notation for the dynamic modeling of transportation problems and defines problem classes and terminology.

3. Problem Description

This section describes the PM operations problem in detail. The description follows the time based flow from the customer sending a package to the point of departure from the origin market. Several small examples are also provided to better explain the operations and flow of packages within the market.

3.1. Supply and Sorting at Stations

On a given day, customers deposit packages at drop off points. Couriers then take these packages to facilities called stations. Packages arrive at stations at fixed times that are considered as incoming *supply points*. Since packages can be of any size, we consider the overall volume of supply as opposed to the number of packages. Incoming supply can be differentiated based on destination and package type such as a box or document. Each unique combination of these identifiers is called a *multi-class package flow*. For easy of discussion, the term package is sometimes used to represent the combination of these identifiers.

Once packages arrive to a station, the supply is loaded onto the *sorting belt*, which is a conveyor belt, and then moved within the facility. We consider two alternative sorting processes at stations and the ramp, proportional and prioritized sorting. Proportional sorting mimics an automated sorting belt where packages pass across the conveyor belt and are sorted automatically into containers. The order that packages are placed on the conveyor belt determines the order in which they are sorted. Prioritized sorting mimics a manual sorting procedure, where package handlers can pick and choose what packages to load into containers.

For both sorting methods, based on a fixed speed of the conveyor belt or number of package handlers, there is a maximum amount of volume that can be sorted in a time period. So if the amount of incoming supply in a time period exceeds the sorting belt capacity, some of the packages will have to be held over to the next time period.

For proportional sorting, while the amount of each multi-class package flow that arrives at a station in a time period is known, it would be difficult to model the layout and the exact order of the packages on the sorting belt. So if the supply at a station that has yet to be sorted exceeds the sort belt capacity, the amount that is sorted is proportional for each multi-class package flow. Then in the next time period, there would be leftover supply from the previous time period as well as the new incoming supply. Since the leftover supply is already queued at the sorting belt, the new incoming supply will be placed at the end of the queue and all leftover supply will be sorted first. So the process can be thought of as a batched first-in first-out (FIFO) queue.

Example of Proportional Sorting: Consider incoming supply of three multi-class package flows at a single station over three time periods and proportional sorting. In the first time period, there is incoming supply of 50 cubic feet of each multi-class package flow. 50 cubic feet of package 1 and 25 of package 2 arrive in the second time period, while 25 cubic feet of each package come to the station in the third time period as shown in Table 1. The maximum sort capacity per time period is 100 cubic feet and sorting begins in the second time period.

Although sorting does not begin until time period 2, the incoming supply in time period 1 is the first to be loaded onto the belt and is first compared with the maximum sort rate. Since the sum of this supply over all multi-class package flows is greater than the sort rate, not all of this supply can be sorted in one time period. For each multi-class package flow, the proportion of supply relative to the total supply of all packages times the sorting capacity will be the amount sorted. This leads to 33.33 cu-ft of each of the three multi-class package flows being sorted in the second time period since this is when the sort begins and 16.67 cu-ft of each package is the leftover supply from time period 1. The incoming supply from time period 2 is also leftover supply since the supply from time period 1 used all of the capacity in the second time period.

In time period 3, the 50 cu-ft of leftover supply from the first time period will be sorted first since it is less than the sort capacity. This leaves the remaining sort capacity of 50 cu-ft to process the next batch of incoming supply, which arrived in time period 2. Since the total supply is greater than the remaining sort capacity, the proportional sorting assumption results in an additional 33.33 cu-ft of multi-class package flow 1 and 16.67 cu-ft of multi-class package flow 2 being sorted in time period 3.

Although there is no incoming supply in time period 4, there is unsorted supply that arrived in time periods 2 and 3 that still must be processed. Based on the batched FIFO queue, the 25 cu-ft of leftover supply from time period 2 will be sorted first. This leaves a remaining sorting capacity of 75 cubic feet as well as 75 cubic feet of supply yet to be sorted. So all of this supply can be sorted and since there is no more scheduled incoming supply, the sort is completed. The

results of proportional sorting can be seen in Table 1. The logic behind prioritized sorting is discussed in Section 3.5. □

Table 1: Proportional Sorting

Multi-class Package Flow	Incoming Supply (cu-ft)	Time	Sorted Supply (cu-ft)
1	50	1	0
2	50	1	0
3	50	1	0
1	50	2	33.33
2	25	2	33.33
3	0	2	33.33
1	25	3	50.00
2	25	3	33.33
3	25	3	16.67
1	0	4	41.67
2	0	4	33.33
3	0	4	25.00

3.2. Assigning Supply to Containers

After determining how much is sorted in a time period, a decision must be made as to where the sorted packages will be placed. A sorting belt consists of a number of locations where volume can be pushed off the sorting belt as seen in Figure 2. At the ramp, there are fixed times when specific multi-class package flows and container sizes are needed in order to depart the market on time. These time, package and container type combinations are called *requirements*, where the container type determines the amount of capacity to be filled.

To fulfill requirements, a pure container with a single multi-class package flow can be created at a station, sent to the ramp, and be directly used to meet a requirement. Alternatively, a multi-class package flow can be combined with other packages at a station to form what is referred to as a *mixed container*. A mixed container must be *broken down* upon arriving at the ramp. This means that at the ramp the volume is removed from the mixed container and resorted to create pure containers that can then be used to fulfill requirements. Any package can be put in a mixed container and it is possible to create a pure container of any multi-class package flow.

Since requirements specify the container type, there is no need to track the volume in pure containers. However, for mixed containers, we need to track the volume of each multi-class package flow since mixed containers are resorted at the ramp. So as volume is being sorted at stations, the decision must be made as to what container type a multi-class package flow will be placed in and whether this will be a pure or mixed container. The process of breaking down a container takes additional time which could result in supply missing its outbound departure, whereas a pure container can be immediately off loaded at the ramp and loaded on outbound vehicles departing the market. However, creating many pure containers can also lead to poor solutions as this could require additional vehicles to transport the containers from the stations to the ramp.

3.3. Container Types and Load Positions

Containers are categorized into two different types, each with varying capacities. The differentiation between these two types is an important factor when considering station to station travel. However, this also plays a role concerning how volume is loaded into containers and how they depart a station.

Refillable Containers

A *refillable* container could be transported to a different station and have additional volume placed in it. At a station, the action of placing volume in a refillable container simply involves moving the designated supply from the sorting belt and dumping it into the container. There can be requirements for refillable containers so therefore it is possible to create both pure and mixed containers of this type.

Shrink Wrapped Containers

For a *shrink wrapped* container, once the decision has been made for the container to depart a station, it is impossible to have more volume placed in this container. While more volume could potentially be placed in the container after it is closed if the container is not already full, this would require additional effort, time and container movement, thus making it operationally infeasible. However, there are also many benefits to creating shrink wrapped containers since they can be moved more easily and effectively.

When a shrink wrapped container is being loaded with volume at a station, it is placed in a *load position*. Load positions are essentially slots or locations that are configured to hold specific sized containers while they are being loaded. Shrink wrapped containers must be placed at load positions to properly close the container whenever it is filled or ready to depart the station.

Since refillable containers need not be at a specific position to be filled, it is assumed that there is no limit to the number of refillable containers that can have volume loaded into them in a time period. However, for shrink wrapped containers, the number of currently loading containers is restricted by the number of containers of each type that can fit in load positions. Also, at least one load position must be reserved for creating a mixed container. This is to prevent all load positions being occupied by pure containers and the arrival of a multi-class package flow that does not have a designated container at a load position. Figure 2 depicts sorting and package movements at a station.

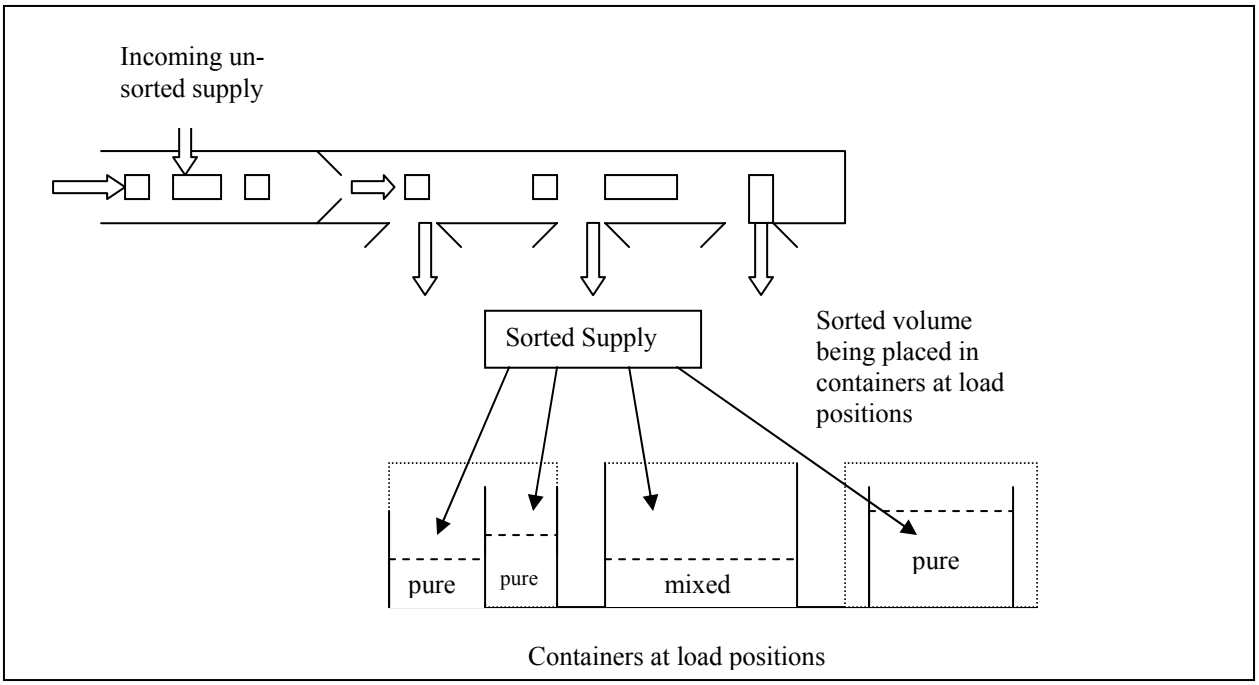

Figure 2: Station Sorting Operations

3.4. Departing Containers and Vehicles

At some point in the daily operations, a container will depart the station. This can be due to the container being filled to capacity, or due to time constraints that require the container to depart the station due to requirements.

Container-Vehicle Relationship

Both refillable and shrink wrapped containers have a maximum volume capacity. Containers are further distinguished by their container footprint. This parameter determines how load positions and vehicles are configured, in a sense taking into account the shape as well as the size of a container. For example, if a load position has a capacity of one, and a certain container size has a footprint of 0.5, two of these containers can be placed at the load position. The logic is similar for containers being placed on vehicles. The sum over the number of containers that depart a station in a given time period times their footprint must be less than the total capacity of the vehicles sent.

3.5. Station to Station Travel

Depending on the overall amount of supply at a station and how supply is assigned, there may be containers or vehicles that are not at capacity. A container being forced out of a station in order to meet a requirement leads to poor resource utilization. After all supply is sorted at a station, having a vehicle travel to another station to pick up additional containers and volume could lead to reduced costs and the need for fewer vehicles across the entire market as shown in Figure 3. Similarly, timing a vehicle to arrive at an intermediate station in time to pick up a container that is forced out to meet requirements can lead to a full vehicle being sent to the ramp as opposed to an additional vehicle being needed just to ship the container that is forced out. This section dis-

cusses how resources are tracked when station to station travel is introduced and describes logical relationships that must occur for station to station travel to be beneficial.

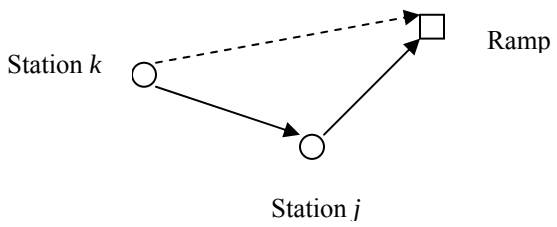

Figure 3: Potential Travel Routes

Shrink Wrapped Containers and Station to Station Travel

By definition, once a shrink wrapped container is closed and departs a station, it cannot be reopened until it is emptied at the ramp or destination market. So clearly, no additional volume could be placed in a shrink wrapped container at an intermediate station. However, it is possible to have a vehicle travel to an intermediate station and pick up a shrink wrapped container that was created from volume that originated at this station. If a container is loaded onto a vehicle that originated at another station, this container is said to *complement* the vehicle. This container could be created before the vehicle arrives, and simply loaded to then depart for the ramp. The number of containers that are at an intermediate station but originated at another station must be tracked in order to ensure that these containers depart for the ramp and so that the vehicle capacity is not violated by the addition of new containers at the intermediate station. Mixed volume from both the origin and intermediate station must be tracked; however, it is not necessary to account for pure shrink wrapped container volume since requirements are in terms of the number of containers as opposed to a specific amount of volume.

Refillable Containers and Station to Station Travel

A refillable container can have additional volume loaded into it before it reaches the ramp. A container must be present at the intermediate station before additional, complementary volume can be loaded into it. So at a station, there is a possibility of assigning incoming supply to complement a refillable container from another station as well as to be assigned to a container that originates at the same station as the supply. The amount of volume in a refillable container from both the origin and intermediate station must be monitored so the amount of volume in a container does not exceed its capacity.

As with shrink wrapped containers, it is also possible to pick up an entire refillable container from an intermediate station and place it on a vehicle from another station. This requires monitoring both containers and volume to ensure vehicle capacity and the departure of all volume to the ramp.

Example: Consider a vehicle originating at station *k* and traveling to station *j*, then to the ramp as shown in Figure 3. Figure 4 represents a snapshot at time *t* when the vehicle departs station *k* with one refillable (*rfc*), and one shrink wrapped (*src)* container.

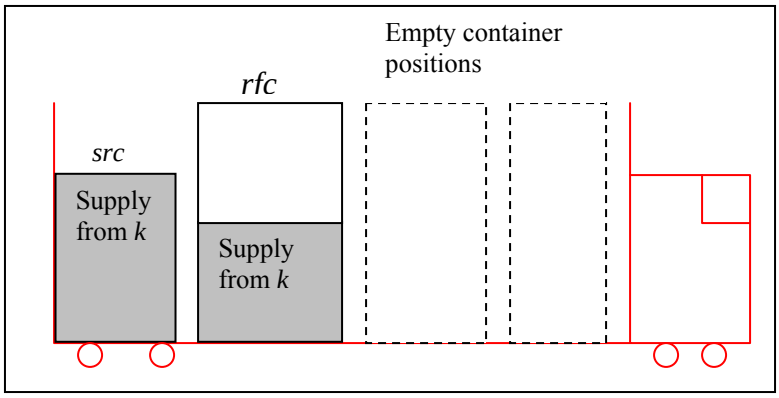

Figure 4: Vehicle Departing the Origin Station

After the vehicle has reached station *j* at time *t* and until it departs station *j*, volume can be assigned to the refillable container, and new refillable or shrink wrapped containers that are created at station *j* can be loaded onto this vehicle before the vehicle departs on its final leg to the ramp. Figure 5 depicts the vehicle at the intermediate station.

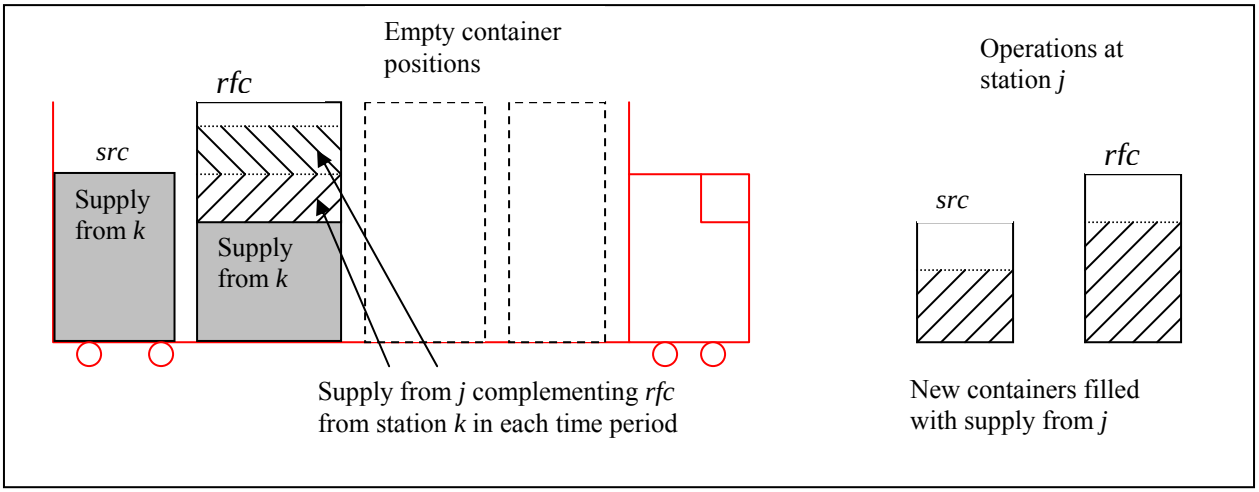

Figure 5: Vehicle at an Intermediate Station

Although the number of containers that originated at station *k* does not change, they still must be accounted for since they are occupying space on the vehicle. It also may be possible for another vehicle to arrive from the same station during this time, thus increasing the number of containers and volume from station *k*. Finally, Figure 6 shows the vehicle at the time it is fully loaded and departs station *j* en route to the ramp. The vehicle holds the shrink wrapped container from *k*, the refillable container from *k* with volume from both stations, as well as the refillable and shrink wrapped containers complementing the vehicle that contain volume originating at the intermediate station *j*. □

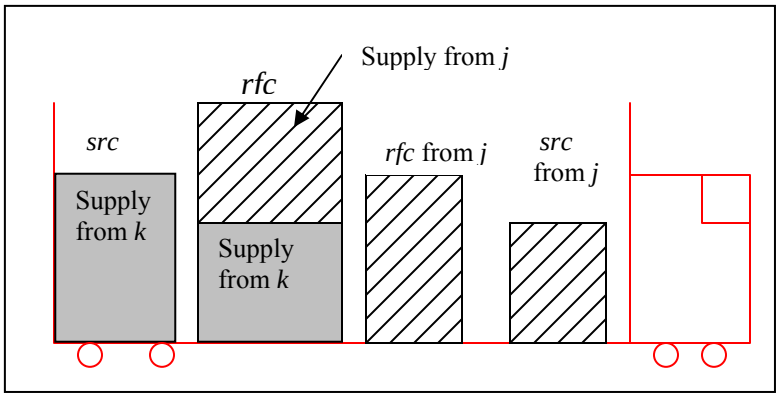

Figure 6: Vehicle Departing the Intermediate Station

Station to Station Numerical Example: This example builds on the examples shown in Section 3.1 and Section 3.5 to show the benefits of stopping at an intermediate station to consolidate volume. Refer to Table 1 for the supply profile at what is denoted as station 1. Now consider another station in the market. For simplicity, we will not discuss sorting at this station and assume the only incoming supply to station 2 is 75 cu-ft of multi-class package flow 1 that is sorted and ready to potentially depart the station in time period 6. The container and vehicle options in Table 2 are available at both stations. Table 3 gives the travel times between the stations and the ramp as well as the requirements at the ramp.

Table 2: Container and Vehicle Parameters

Travel Time	2 time periods
Ramp Sort Rate	100 cu-ft/time period
Ramp Sort Start	1st time period
Vehicle 1 Footprint	1
Vehicle 2 Footprint	2.5
Container 1 Capacity	100 cu-ft
Container 1 Footprint	0.5
Container 2 Capacity	300 cu-ft
Container 2 Footprint	1
Vehicle 1 Cost	$400.00
Vehicle 2 Cost	$600.00
Ramp Sort Cost	$1.00 per cu-ft

Table 3: Requirements (right) and Travel Times (left)

	Station 1	Station 2	Ramp
Station 1	-	2	2
Station 2	2	-	1

Multi-class Package Flow	Containers Required	Container Type	Time
1	3	1	8
2	1	1	9
3	1	1	8

The solution for the single station problem results in sending four pure containers on vehicle 2. Note that the footprint capacity of vehicle 2 is 2.5 while the sum of the footprints of the containers on this vehicle is 2, meaning that there is room on the vehicle for another container with

footprint 0.5. The optimal solution to the two station problem without allowing station to station travel would result in sending vehicle 1 from station 2 to the ramp with a pure container of multi-class package flow 1 as well as the previously discussed vehicle from station 2. This results in a total cost of $1000.

With station to station travel, the vehicle that departed station 2 can be eliminated and the pure container of multi-class package flow 1 at station 2 can be used to complement the vehicle that originated at station 1. The vehicle from station 1 will once again depart in time period 4 with 4 pure containers and an additional spare 0.5 footprint capacity. Instead of traveling directly to the ramp, it takes two time periods to travel to station 2, thus arriving by time period 7. The pure container of multi-class package flow 1 that has already been created can be loaded onto this vehicle, and then travel to the ramp in one time period, allowing all containers with a time of 8 to meet requirements since they are pure containers and there is no need for resorting. This results in a solution with a significantly reduced cost of $600 since only one vehicle is needed for two stations. If time permitted and there were requirements for refillable containers, the 75 cu-ft of multi-class package flow 1 at station 2 could complement the pure containers that originated at station 1 as opposed to creating a new container. □

4. Dynamic Program Model Description

This section describes the resources that must be tracked in the dynamic programming model. We then provide a numerical example to describe how the states are updated, some of the constraints on the actions in the system, and give an example of some system dynamics and action space constraints when considering station to station travel. The full mathematical model is very complex and can be found in Schenk (2005).

4.1. States

Due to the complexity, we employ the modeling framework from Powell et al. (2001), which is based on the notion of resources and attributes.

From the point of the incoming supply at stations, several resources must be tracked to ensure that all supply eventually departs the origin market and the decisions made in a time period are based on the appropriate resources that are present. Resources are represented by an attribute vector a. We have already indicated that for each container we must distinguish between a pure or a mixed one and therefore this is the first attribute. Containers differ in their size and shape and therefore for each container the second attribute represents its type. The underlying multi-class package flow is also an attribute. Since containers also move between locations on vehicles, the last three attributes correspond to the origin and destination station, and the underlying vehicle. When referring to containers stationed at a station, we specify the origin station attribute and leave the last two attributes empty.

In order to appropriately load containers into vehicles we need to track the number of containers or resources of each attribute vector. On the other hand, to capture requirements for each mixed container we need to know the volume of each container or resource of each attribute vector. This requires two type of state variables; one for counting the number of resources and the other one to represent the underlying volume.

Let a = (‘pure’ or ‘mixed’, container type, multi-class package flow index, station index, station index, vehicle type) be the attribute vector. Its coordinates are labeled as $a_1, a_2, ..., a_6$. The

two station indices capture the origin and destination station indices of a potential transfer carried out by the corresponding vehicle. If an attribute is not needed for a particular state component, we represent it with an underscore.

We denote by r_{ta} the amount of volume at time *t* in all containers with attribute vector *a*. For example, if $a = (\text{'mixed'}, s, i, j, \_, \_)$, then r_{ta} denotes the amount of volume of multi-class package flow *i* in mixed containers of type *s* at time *t* and station *j* that are ready to leave the station. If type *s* is a shrink wrapped container, then these containers have already been removed from load positions. There are similar state variables $\hat{r}_{ta}$ that capture the number of containers. The amount of volume in containers at load positions must be tracked to determine when containers are full and should be removed. As an example, $\bar{r}_{ta}$ with $a = (\text{'pure'}, s, i, j, \_, \_)$ has the same interpretation as r_{ta} except that it is the volume in a pure container *s* at a load position. Once removed, we need to know the number of available containers and amount of volume that could potentially depart the station in each time period. Since requirements at the ramp are given in terms of containers rather than volume, we do not need to know the amount of pure volume ready to depart, simply the number of containers. For mixed containers, the volume of each multi-class package flow is important so we can guarantee that volume departs the station in order to be resorted at the ramp.

Due to the limited sort capacity in a time period we must allow for backlogging of incoming supply at stations and incoming volume at the ramp. In other words, volume that is not yet sorted in a time period carries over to the next time period. To model this we introduce r_{ta}^{NS} with $a = (\_, \_, i, j, \_, \_)$, which represents the amount of volume of multi-class package flow *i* that is at station *j* or the ramp if *j* corresponds to the ramp before time *t* but has not yet been sorted.

When the decision is made to have containers depart stations, we need to know the time that they depart the station as well as the time that they arrive at the ramp since this will not necessarily be the next time period due to multi period travel times. We let time *t* be the time that containers or volume depart stations and time t' be the time that these resources arrive at the ramp. This information is then used to determine the amount of volume that must be resorted and the number of pure containers available to meet requirements. As an example, consider $\hat{r}_{tt'a}$ with $a = (\text{'pure'}, s, i, \_, \_, \_)$, which is the number of pure containers of multi-class package flow *i* type *s* known at time *t* and arriving at time t' to the ramp. Although these containers and the corresponding volume are not present at the ramp at time *t,* actions are taken at this time that predetermine the state of the system after the multi period travel time to the ramp. Once supply is resorted, we need to track this amount to know what volume is available to potentially be used to meet requirements. For example, $\tilde{r}_{tt'a}$ with $a = (\text{'mixed'}, \_, i, \_, \_, \_)$ tracks the amount of multi-class package flow *i* arriving to the ramp at time t' in mixed containers.

Next we outline states that are required to capture the new features of the problem presented herein. Concerning volume at load positions, this model tracks the time since a container of a given type and multi-class package flow has been removed from a load position to capture the switching time to remove and then place a new container at a load position. We must assure that sufficient time is elapsed since the removal of a container of a particular type from a load position before we can start using a new container at a load position. We need, for example, r'_{ta} with $a = (\text{'pure'}, s, i, j, \_, \_)$ to represent the number of time periods since a pure container of type *s* and multi-class package flow *i* has been removed from a load position at station *j*. If this value is

less than a given parameter, then we cannot load a new container at a load position. Whenever a container with these characteristics is removed from a load position, this value is set to one and then it is increased by one each subsequent time period.

We must also account for containers and volume traveling between stations. Consider the following four possibilities of station to station movement.

1. A pure shrink wrapped container (*src*)
2. A pure refillable container (*rfc*)
3. A mixed shrink wrapped container (*src*)
4. A mixed refillable container (*rfc*)

For all four cases, the number of containers as well as the time that these containers arrive at the intermediate station must be tracked to account for the space that they are occupying on the vehicle. The difference between the cases concerns volume. For case 1, the volume in the container does not need to be tracked since it cannot acquire additional volume and will be used directly to meet requirements. For a pure refillable container, we must account for the volume arriving to the intermediate station to ensure that the container capacity constraint is not violated by adding complementary volume into the container. The volume in cases 3 and 4 must clearly be monitored since this volume will be resorted at the ramp after departing the intermediate station.

Now consider the containers in all four cases and volume for the cases where this must be tracked once the vehicle has arrived at the intermediate station. New states are needed to track containers and volume from the origin station at the intermediate station since this information is not conveyed in the states linking the departure and arrival times of the two stations. Tracking containers and volume in limbo at the intermediate station is necessary to guarantee that these resources eventually depart for the ramp and that the vehicle capacity is not violated by additional containers and supply from the intermediate station. For example, we denote by $\hat{r}_{ta}$ with $a = (\text{'pure'}, s, i, k, j, l)$ the number of pure containers of type *s* multi-class package flow *i* in vehicle type *l* that are at station *j* at time *t* and arrived from station *k*.

An advantage of this modeling paradigm is that each state coordinate can be encoded by at most three indices, the time index *t*, the actionable time t', and the underlying attribute vector *a*.

Example: Here we follow the example shown in Section 3.6 of a vehicle departing station *k*=1 with a refillable and shrink wrapped container and traveling to station *j*=2 to pick up complementary volume and containers before proceeding to the ramp. The parameters used are different from those given in Table 2 and Table 3. The example presented in Section 3.6 was given to show the potential benefits of station to station travel while the example given next is intended to show how states are tracked across the time horizon. Let *src* in Figure 4 be a pure container with attribute vector ('pure',1,1,1,2,1), and let the current time period be *t*=10 with the travel time from station *k* to station *j* being 10 time periods. We assume *rfc* represents a different container with attributes ('pure',2,2,1,2,1). Containers of type 1 are shrink wrapped while those of type 2 are refillable. Also let there be 50 cu-ft of volume in *rfc*. Clearly the action variables representing the departing containers will equal one at time *t*=10. Assuming that there are no other available containers on hand at station 1, the number of available containers in the next time period will be driven to zero by the departing vehicle. Since we must also track the time when containers or volume arrive to the intermediate station, state variables are needed to

represent that a container departing in time period 10 will arrive in time period 20 after the travel time of 10 time periods.

The tracking of containers and volume across multi period travel times is necessary to determine when the resources will arrive at the intermediate station as well as to monitor the resources that originated at station *k* while they are in limbo at station *j*=2 to guarantee that they eventually depart for the ramp. Consider time period $t = 20$ when the vehicle arrives at station *j* and assume that no volume or complementary containers have been added to the vehicle. For the refillable container, we must track that at this time period a container arrives as well as the 50 cu-ft of volume in the container.

While the vehicle is in limbo at station *j*=2, *rfc* can have additional volume loaded into it in each time period and containers of both types can be created at *j* to complement the vehicle in a later time period as depicted in Figure 5. Consider time period *t*=21 after the vehicle has arrived and assume that the vehicle will not depart for the ramp in this time period. Suppose that it is decided to have 10 cu-ft of multi-class package flow *i*=2 be loaded into the *rfc* that arrived from station 1 in the previous time period. Assume that there are no additional vehicles arriving to station 2 in this time period or throughout the rest of the time horizon. The total amount of volume from either station that is in *rfc* will then consist of the 50 cu-ft that originated at station *k*=1 as well as the 10 cu-ft that have just been assigned to this container.

Now let *t*=30 represent the time when the vehicle that originated at station 1 departs station 2 for the ramp. Clearly the complementary containers that were created at station 2 must have been loaded onto the vehicle by this time and volume can no longer be added to *rfc*. Suppose that during the time that this vehicle was in limbo at station 2, 50 cu-ft of supply was loaded into an *rfc* that originated at station 1. State variables concerning the number of refillable and shrink wrapped containers as well as the amount of volume in the refillable container on the vehicle that originated at station 1 will be driven to zero in the next time period due to the departing resources. Other states such as those representing the *src* and the volume and mixed containers complementing this vehicle will also drop to zero in the time period after the vehicle departs for the ramp. These departing containers and volume must be tracked to determine when the resources will arrive at the ramp. □

4.2. Sample System Dynamics

Here we provide a sample of the system dynamics that pertain specifically to station to station travel. Note that only those parameters and variables that are used in the following system dynamics are formally defined. The next section gives a similar sample of the action space constraints that are relevant for station to station travel. For a complete description of the dynamic program, refer to Schenk (2005).

Let *D* be the set of all decision types. These decision types act on resources and the corresponding attribute vector. For example, the decision of removing a container from a load position is encoded as *d*='rmv' $\in D$.

Let $\hat{r}_{ta}, a = (\_, s, i, j, \_, \_)$ be the state representing the number of refillable or shrink wrapped containers of type *s* containing the pure or mixed multi-class package flow *i* at station *j* that have been removed from load positions and could potentially depart the station at time *t*. Also, let us define the variable $x_{tda}, a = (\_, s, i, j, \_, \_), d =$ 'rmv' to be the number of refillable containers of type *s*, multi-class package flow *i* removed from a load position at station *j* at the end of time period *t*. In our case it can only be 0 or 1 since we have at most one container of type *s* in all load

positions. Similarly, we define $x_{tda}, a = (\_, s, i, j, k, l), d =$ 'trv' to be the number of containers of type *s* of multi-class package flow *i* sent from station *j* to another station or the ramp *k* at time *t* on vehicle *l*, and let $x_{tda}, a = (\_, s, i, k, j, l), d =$ 'cmp' represent the number of containers of multi-class package flow *i* container type *s* created at station *j* and used in time *t* to complement vehicle *l* that originated at a different station *k*. The following equation tracks the number of pure and mixed volume containers on hand at a station that are ready to depart.

$$\hat{r}_{t+1,a} = \hat{r}_{ta} + x_{t,\text{'rmv'},a} + \sum_{\bar{a}} \sum_{d \in \{\text{'trv'},\text{'cmp'}\}} \sigma_a(\bar{a}, d) x_{td\bar{a}}$$

$$\sigma_a(\bar{a}, \text{'trv'}) = \begin{cases} -1 & \bar{a} = (a_1, a_2, a_3, a_4, \bar{a}_5, \bar{a}_6) \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_a(\bar{a}, \text{'cmp'}) = \begin{cases} -1 & \bar{a} = (a_1, a_2, a_3, \bar{a}_5, a_4, \bar{a}_6) \\ 0 & \text{otherwise} \end{cases}$$

This equation shows that a container may leave the station on a vehicle that originated at this station, or complement a vehicle that arrived to station *j* from another station. The summation over all attributes $\bar{a}$ corresponds to the summation over all vehicle types and all stations. There is such a system dynamics equation for each $a = (\_, s, i, j, \_, \_)$.

Another extension to the model presented in Schenk and Klabjan (2005) concerns the switching time between removing a container from a load position and placing a new container in the position. The state variable r'_{ta} was already defined in Section 3.1. This information is used in action space constraints to ensure that a container of the same multi-class package flow and type is not assigned volume until sufficient time has passed. By definition we have

$$r'_{t+1,a} = r'_{ta} \cdot \left(1 - x_{t,\text{'rmv'},a}\right) + 1$$

for each $a = (\text{'pure'}, s, i, j, \_, \_)$.

In general, each system dynamic equation can be written in the form

$$R_{t+1,a} = R_{ta} + \sum_{\bar{a}} \sum_{d \in D_1} \sigma_a(\bar{a}, d) x_{td\bar{a}} \pm R_{ta} \cdot x_{t\bar{d}a} + \tau_{ta}$$

for some function σ_a, constant τ_{ta}, and decision type $\bar{d}$. In addition, $x_{t\bar{d}a}$ is binary. Here *R* represent any one of the different parts of the state.

4.3. Action Space Constraints

This section provides a description of selected action space constraints that concern station to station traffic and other extensions to the intra market optimization problem that are discussed in this paper. Any variables or parameters that are needed to express the action space constraints that have not already been explicitly defined are presented in this section.

For vehicles that depart an intermediate station for the ramp, the containers on this vehicle include those originating at another station as well as any containers complementing this vehicle that were picked up at the intermediate station. Binary variable $x_{tda}, a = (\_, \_, \_, k, j, l), d =$ 'snd' is 1 if a vehicle of type *l* that originated at station *k* departs intermediate station *j* at time *t*. Vari-

able $x_{tda}, a = (\_, s, i, k, j, l), d =$ 'crt' is used to represent those containers that were created at the origin station *k* and are also on the vehicle. The vehicle capacity requirement reads

$$\sum_{\bar{a}} \sum_{d \in \{'\text{cmp}','\text{crt}'\}} e_{a_1} \sigma_a(\bar{a}, d) x_{tda} - u_{a_6} \cdot x_{t,'\text{snd}',a} \le 0$$

$$\sigma_a(\bar{a}, '\text{cmp}') = \sigma_a(\bar{a}, '\text{crt}') = \begin{cases} 1 & \bar{a} = (a_1, \bar{a}_2, \bar{a}_3, a_4, a_5, a_6) \\ 0 & \text{otherwise.} \end{cases}$$

Here e_s is the footprint of container type *s* and u_l is the footprint capacity of vehicle type *l*. Note that the summation is over all container types and multi-class package flows. These constraints are present for each attribute vector $a = (\_, \_, \_, k, j, l)$.

Constraint (1) below tracks when containers are removed from load positions, and guarantees that no volume of a given package and container type is assigned until sufficient time has elapsed to unload, and then load a new container into a load position. In action space constraint (1), x_{tda}, $a = ('\text{mixed}', s, i, j, \_, \_)$, *d*='asg' represents the amount of multi-class package flow *i* at station *j* that is assigned to a mixed container of type *s* at time *t*. Quantities v_s and $ul_{ta}, a = (\_, s, i, j, \_, \_)$ are input parameters representing the capacity of container *s* and the time needed to switch a container of multi-class package flow *i* type *s* at station *j*, respectively. Thus we have

$$x_{t,'\text{asg}',a} \le v_{a_2} \left(r'_{ta} - u_{ta} \right)^+ \tag{1}$$

for each attribute vector *a* = ('mixed',*s*,*i*,*j*,_,_).

A further example of including the option of station to station traffic in the dynamic programming model concerns assigning supply to containers. All volume that is sorted in a time period must be assigned to a container type. Volume of a multi-class package flow may be assigned to a container that is created at the same station as the incoming supply or to complement a refillable container from another station. Variable $x_{tda}, a = (c, s, i, k, j, l), d =$ 'cmp-r' represents the amount of a given multi-class package flow assigned to complement a refillable container that is currently at station *j* but that originated at a different station *k*. We have

$$\sum_{\bar{a}} \sum_{d \in \{'\text{arg}','\text{cmp-r}'\}} \sigma_a(\bar{a}, d) x_{tda} \le x_{t,'\text{srt}',a}$$

$$\sigma_a(\bar{a}, '\text{arg}') = \begin{cases} 1 & \bar{a} = (\bar{a}_1, \bar{a}_2, a_3, a_4, a_5, a_6) \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_a(\bar{a}, '\text{cmp-r}') = \begin{cases} 1 & \bar{a} = (\bar{a}_1, \bar{a}_2, a_3, a_4, \bar{a}_5, a_6) \\ 0 & \text{otherwise} \end{cases}$$

We denote by $x_{t,'\text{srt}',a}$ the maximum amount of multi-class package flow a_3 that can be sorted in time *t* at station a_4. Here the summation is over mixed and pure attributes, container types, and destination stations. There is such a constraint for each $a = (\_, \_, i, j, \_, \_)$.

Next we discuss proportional sorting. Here we briefly formalize the concepts introduced in Section 3.1. For simplicity we consider only the ramp although the treatment of a station is very similar. Let $w_{ta}, a = (\_, \_, i, ra, \_, \_)$ be the amount of incoming supply of multi-class package flow *i* to the ramp in time period *t* and let $c_{t,ra}$ be a parameter representing the maximum sorting belt

rate at the ramp. Here *ra* represents the ramp. By definition from Section 4.3 it follows that $\tilde{r}_{tta}, a = (\text{'mixed'}, \_, i, ra, \_, \_)$ is the amount of multi-class package flow *i* that has just arrived to the ramp in mixed containers. Expression (2) gives the sorted volume of each multi-class package flow. The first term on the right-hand side expresses the total amount of multi-class package flow *i* that is yet to be sorted. In case the capacity is not binding it represents the amount that is sorted. The second term in (2) gives a proportion of each multi-class package flow. The denominator represents the total volume that is yet to be sorted. The numerator then takes the total amount of yet to be sorted multi-class package flow *i* and the corresponding sort capacity. Note that (2) is nonlinear in the state space variables.

$$x_{t,\text{'str'},a} = \min\left\{ r_{ta}^{NS} + \tilde{r}_{t,t,s(a)} + w_{ta}, \frac{(r_{ta}^{NS} + \tilde{r}_{t,t,s(a)} + w_{ta}) \cdot c_{t,ra}}{\sum_{\bar{a}} \sigma_a(\bar{a}, \_)\left(r_{t\bar{a}}^{NS} + \tilde{r}_{t,t,s(\bar{a})} + w_{t\bar{a}}\right)} \right\}$$

$$\sigma_a(\bar{a}, \_) = \begin{cases} 1 & \bar{a} = (a_1, a_2, \bar{a}_3, a_4, a_5, a_6) \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

$$s(a) = (\text{'mixed'}, a_2, a_3, a_4, a_5, a_6)$$

There is such a constraint for each *a* = (_,_,*i*,*ra*,_,_).

It turns out that each action space constraint can be written in the form

$$\sum_{\bar{a}} \sum_{d \in D_1} \sigma_a(\bar{a}, d) x_{td\bar{a}} \le f_{ta}(R_{ta})$$

for some function f_{ta}.

5. Solution Methodology

This section gives the general approach used to solve the dynamic program with station to station traffic and proportional sorting. Topics concerning station to station travel and proportional sorting are explicitly mentioned while the reader is referred to Schenk (2005) for a discussion of further details of the solution methodology.

5.1. Approximate Dynamic Programming

The methodology used to solve the dynamic programming model with station to station travel is based on the stochastic gradient algorithm for approximate dynamic programming, see e.g. Powell and Van Roy (2004) and Powell et al. (2003). Even though the dynamic program developed to solve the intra market operations problem is deterministic, to comply with the existing terminology and literature on the subject, we call the presented algorithm the stochastic gradient algorithm. We use a linear approximation of the value function since this was shown to work well for problems with multi period travel times in Powell and Carvalho (1998) and provides a computationally tractable model.

Let R_t represent the vector of all states at time *t*, let V_t be the value of being in a particular state at time *t*, and let c_t and X_t represent the respective single period cost function and action variables. Then the optimality equation is written as

$$V_t(R_t) = \min_{X_t}\left(c_t\left(X_t, R_t\right) + V_{t+1}\left(R_{t+1}\right)\right).$$

In our case, the initial value of all states R_0 is zero. Since it is desired to have all supply depart the market, we want the final value of all states at the end of the time horizon to be also zero. Since we are minimizing the value function, a large penalty is imposed for having non zero state values in the last time period *T*, that is $V_T(R_T) = \infty$ for $R_T > 0$ and $V_T(0) = 0$.

Standard discrete dynamic programming techniques cannot be applied to the value function recursion due to the large state space (curse of dimensionality) and heavily constrained actions. In approximate dynamic programming the value function $V_t(R_t)$ is approximated by $\widetilde{V}_t(R_t)$. Each coordinate of the state is indexed by the attribute vector *a* and/or t'. When t' is not present, we can assume that $t' = t$ and thus we suppose that $R_t = (R_{tt'a})_{t' \geq t, a}$. We use a linear approximation and therefore $\tilde{V}_t(R_t) = \sum_a \sum_{t' \geq t} \alpha_{tt'a} \cdot R_{tt'a}$. The whole problem now transforms into the one of finding good coefficients α. We define

$$\hat{V}_t(R_t) = \min_{X_t} \left(c_t(X_t, R_t) + \widetilde{V}_{t+1}(R_{t+1}) \right), \tag{3}$$

where $\hat{V}_t(R_t)$ is merely a placeholder. In the algorithm $\hat{V}_t(R_t)$ needs to be computed several times for fixed states R_t.

Observe that $\alpha_{tt'a} = \dfrac{\partial \tilde{V}_t}{\partial R_{tt'a}}$ and therefore α's are updated based on partial derivatives of $\hat{V}_t(R_t)$. Since this function is not differentiable at some points and it is also neither convex nor concave, we consider "local subgradients". In our case it can be shown that (3) is equivalent to

$$\hat{V}_t(R_t) = \min \quad \bar{c}_t \cdot \bar{X}_t$$

$$A\bar{X}_t \leq f(R_t).$$

Some parts in (3) need to be linearized and we denote by $\bar{X}_t$ the expanded actions. Each term $R_{tt'a} x_{t\hat{a}a}$ is replaced by $\bar{x}_{t\hat{a}a} \leq R_{tt'a}, \bar{x}_{t\hat{a}a} \leq M \cdot x_{t\hat{a}a}$ for a large enough number *M*. Clearly for this linearization to be valid the objective coefficient must be negative. It turns out that this is always the case. Some of the actions are required to be binary but in subgradient computation we use the linear programming relaxation.

Assuming that the functions *f* and $\hat{V}_t$ are differentiable, we define the subgradient as follows where *m* represents the number of rows in the constraint matrix *A*.

$$\Pi_{tt'a} = \frac{\partial \hat{V}_t}{\partial R_{tt'a}} = \sum_{p=1}^{m} \pi_{tp} \cdot \frac{\partial f_p}{\partial R_{tt'a}} \tag{4}$$

In (4), $f = (f_1, ..., f_m)$ and π_t is an optimal dual solution. Even though $\hat{V}_t$ is not necessarily differentiable, we still use (4) to approximate a descending direction.

Based on (4) and the dual variables, update equations are generated for all state variables. Refer to Schenk (2005) for detailed examples of these equations. Next we address proportional sorting. Recall (2), which defines the amount to be sorted. When solving (3), observe that the states are given, which implies that (2) is not really a constraint but it specifies some actions.

Since the right-hand side of (2) is not differentiable, we approximate the subgradient by considering which of the two values represents the minimum. The following formula can be found for example in Bertsekas (2003). Let us denote the minimization of the two functions by $\bar{h}(x) = \min\{h_1(x), h_2(x)\}$ and let $h_1'(x)$ and $h_2'(x)$ represent the derivatives of the two functions. We compute the gradient as follows.

$$\nabla \bar{h}(x) = \begin{cases} h_1'(x) & \text{if } h_1(x) < h_2(x) \\ h_2'(x) & \text{if } h_1(x) > h_2(x) \\ \lambda \cdot h_1'(x) + (1-\lambda) \cdot h_2'(x) & \text{if } h_1(x) = h_2(x) \text{ and any } \lambda \in [0,1] \end{cases}$$

In our case, h, h_1 and h_2 are functions of two variables, r_{ta}^{NS} and $\tilde{r}_{tta}$, and therefore the derivatives are replaced by partial derivatives. If the third case occurs, we compute λ in such a way that we get the steepest decent. Formally, we minimize $\left\| \lambda \cdot h_1'(x) + (1-\lambda) \cdot h_2'(x) \right\|_2$.

Because the gradient can fluctuate, the following smoothing equation (5) is used to give weights to gradient information at the current iteration as well as at the previous approximations.

$$\alpha_t^n = (1-\lambda^n) \cdot \alpha_t^{n-1} + \lambda^n \cdot \Pi_t^n \tag{5}$$

In (5), α_t^n is the value function approximation in iteration *n* and time *t* and $\Pi_t^n = \left(\Pi_{tt'a}^n\right)_{t' \geq t, a}$. The smoothing constant $\lambda^n \in (0,1)$ can be adjusted throughout the algorithm to help speed convergence.

5.2. Algorithm Summary

Algorithm 1 gives the main steps of the approximate dynamic programming methodology.

Step 0 Initialization: Choose an approximation $\widetilde{V}_t^1$ for V_t^1 for all *t*. Set iteration counter $n = 1$.

Step 1 Forward Pass:

Step 1.1 Initialize forward pass: Set $t = 1$. Initialize R_t

Step 1.2 Solve the myopic problem: For time period *t* solve the approximate myopic problem (3) by using $\widetilde{V}_{t+1}^n$ subject to the action space constraints to get X_t.

Step 1.3 Apply the system dynamics: Calculate R_{t+1}.

Step 1.4 Advance time: Set $t = t + 1$. If $t < T$ go to Step 1.2.

Step 2 Advance iteration counter: Set $n = n + 1$.

Step 3 Value function update: Calculate $\alpha_t^n = (1-\lambda^n) \cdot \alpha_t^{n-1} + \lambda^n \cdot \Pi_t^n$ for all $t = 1, \ldots, T$ to provide an approximation $\widetilde{V}_t^n$ for the next iteration. Go to Step 1.

Algorithm 1: The Approximate Dynamic Programming Algorithm

6. Computational Experiments

In this section, we compare the dynamic programming model discussed in this paper, the dynamic programming model discussed in Schenk and Klabjan (2005), the integer programming model given in Schenk (2005), and current operations in practice. The main interest of the company was in evaluating potential benefits of the station to station travel, which is not considered in the current practice. For this reason we kept switching times and the sorting process identical in all experiments. Since most of the stations use manual sorting, the sorting process at stations is modeled as prioritized sorting and at the ramp we use proportional sorting. The dynamic program in Schenk and Klabjan (2008) does not allow for station to station traffic. It also does not capture container switching times at load positions and proportional sorting but to focus solely on the station to station traffic we have added these two features to the model.

The models were tested on two different markets. Table 4 provides details about the range of the size of parameters for both market sizes, which will be referred to as M1 and M2 respectively. The main parameters that differ between M1 and M2 are the number of stations, time horizon, and the overall incoming volume of supply to the market

Table 4: Market Parameters

	M1	M2
Number of Stations	4-6	9-13
Number of Multi-class Package Flows	22-26	22-26
Number of Container Types	5-7	7-9
Number of Vehicle Types	5-7	7-9
Number of Time Periods	250-350	350-500
Total Supply	14,000-18,000 cu-ft	21,000-25,000 cu-ft

6.1. Results

Computational experiments were performed on a Dell PC with a Pentium® 4 1.80GHz processor, 1GB of RAM, and Windows XP operating system. Microsoft Visual C++ 6.0 was the development environment and we used ILOG CPLEX 8.0 with Concert Technology 1.0 as the mixed integer programming solver.

M1 Results

The following table gives solution characteristics for M1 using the four solution methods. DP1 refers to the direct station to ramp dynamic program given in Schenk and Klabjan (2008) while DP2 includes station to station traffic. MIP refers to the integer programming formulation of the PM operations problem while the fourth row gives results based on the current practice. Due to confidentiality reasons, we cannot discuss further details of the current operational practices.

Table 5: M1 Solution Characteristics

	Objective	Run Time (hours)	Time per Iteration (mins)	No. of Vehicles Used	Pure Vol.	Service Level	Vehicle Capacity Utilization	Container Capacity Utilization
DP1	106.9%	14	15	9	41%	100.0%	95.83%	80.86%
DP2	100.0%	16	17	8	41%	100.0%	95.83%	80.86%
MIP	136.9%	120	-	15	0%	100.0%	-	-
Current	120.2%	-	-	11	57%	99.9%	97.20%	61.49%

Objective values are given as the percentage gap from the best solution obtained, which for this instance is DP2. It can be seen that DP2 outperforms the current operations in practice by over 20 percent. Cost savings are incurred by considering aggregating volume at stations compared with direct station to ramp routing. The MIP formulation presented in Schenk (2005) was modified considerably in order to even obtain an integer solution, let alone one that is close to the LP relaxation value. Only direct station to ramp paths were considered and pure volume was not allowed as the addition of constraints and variables to incorporate pure volume and containers more than doubles the size of the model. Even with this greatly restricted mixed integer program, it took nearly 24 hours to find the first integer solution and after running for three more days there was still a considerable IP/LP gap. It is unlikely that allowing pure containers in the MIP would lead to comparable or better solutions than the dynamic programs even if this were tractable since the cost difference is due primarily to almost double the number of vehicles being used as opposed to just the additional sorting cost of mixed volume.

Comparing the current operations with DP1 and DP2, we see that the dynamic programs result in a lower cost because fewer vehicles are used. Although the current practice sends more pure containers and there is less volume to be resorted, it requires more vehicles and space is not used as efficiently. Current practice results in containers with less volume, thus causing wasted space and more vehicles to transport the volume. Service level is the percentage of volume that is able to depart the market on time. As seen in Table 5, all volume is sorted in DP1 and DP2 while a small amount of volume remains at the ramp in practice. Although more volume must be resorted at the ramp, it is sent early enough that there is sufficient time to resort all of the mixed volume.

There are predetermined times when the sorting process begins at stations and the ramp as well as implied ending times after all volume has been sorted. Vehicles leaving a station before the implied ending time are often forced out to meet a requirement. This implies that such vehicles have no extra time to visit an intermediate station. For this reason to improve the convergence of the algorithm we did not allow for station to station travel before the implied ending times. After the ending times several vehicles depart for the ramp and many of them are usually not at full capacity. Vehicles can now potentially be consolidated. The approximate dynamic programming algorithm found a solution that resulted in one less vehicle for model DP2 compared with DP1 and the savings of an entire vehicle outweighed the additional costs associated with the multi leg travel route.

Figure 7 shows the objective value after each full pass of the dynamic programming algorithm for DP2. The initial value function approximation along with limiting the possibilities of station to station travel yields a reasonable initial solution as nearly all of the supply departs the ramp and the solution is within 15 percent of the best solution obtained at a later iteration of the

algorithm. Figure 8 gives the service level per iteration for DP2. Initially the service level is beyond 100% but quickly it reaches the optimal level.

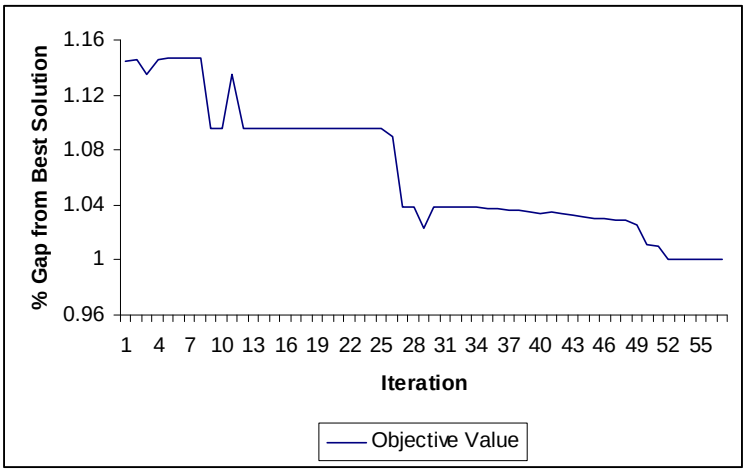

Figure 7: Objective Improvements for Market M1

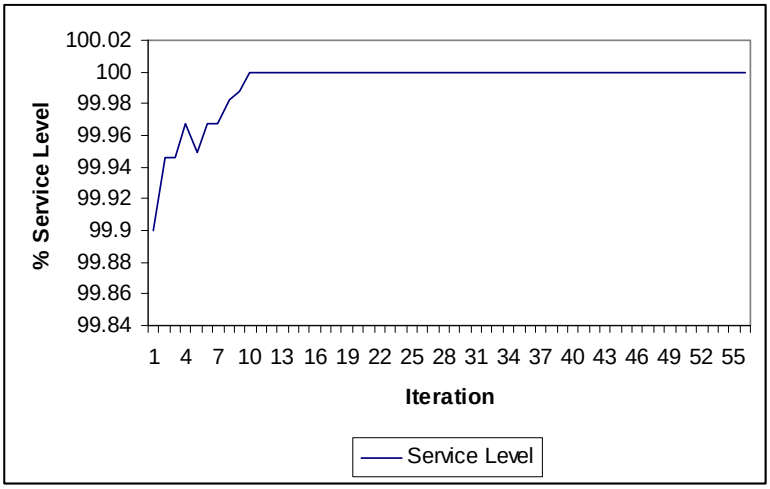

Figure 8: Service Level Improvements for Market M1

M2 Results

Table 6 provides the results for the larger market modeled by both dynamic programs and the current practice. We were not able to obtain a feasible solution to the integer program.

Table 6: M2 Solution Characteristics

	Objective	Run Time (hours)	Time per Iteration (mins)	No. of Vehicles Used	Pure Vol.	Ser-vice Level	Vehicle Capacity Utilization	Container Capacity Utilization
DP1	104.3%	24	144	20	67%	99.10%	87.72%	69.75%
DP2	100.0%	24	144	19	67%	99.10%	91.14%	82.69%
Current	133.3%	-	-	32	71%	98.90%	100.00%	38.72%

The results for the larger market size also lead to the conclusion that considering station to station travel in the approximate dynamic program leads to improved costs over both the original dynamic program without allowing travel to intermediate stations and current operational practice. DP2 outperformed the current operations by over 30 percent. The reason for this improvement was also consistent with the results from market M1. While the dynamic programs send less volume in pure containers than current practice, the containers that are created are filled to a greater level thus resulting in the need for fewer vehicles to transport the volume. DP2 resulted in a significantly smaller number of vehicles than current practice and considering station to station travel saved the cost of one vehicle and resulted in improved container and vehicle capacity utilization compared to the best solution obtained from DP1.

Due to the problem size, run times were considerably longer for M2 compared with M1. Trials were stopped after 24 hours of run time for DP1 and DP2 and the best solution is reported. While M2 is a larger data instance than M1, the main factor that causes an increase in run time is the number of time periods in the time horizon. As the number of time periods increases, the number of states and approximation values that must be generated increases dramatically due to states where both actionable time t and the effect time t' must be accounted for.

Figure 9 and Figure 10 give the objective value and service level per iteration for DP2, respectively. Although there are less iteration compared to the experiments with the smaller market M1, there is a clear improving trend in the objective value and service level. Initial value function approximations lead to reasonable initial solutions and service levels. Longer running times and further experimentation with updating the smoothing parameter could lead to improved solutions.

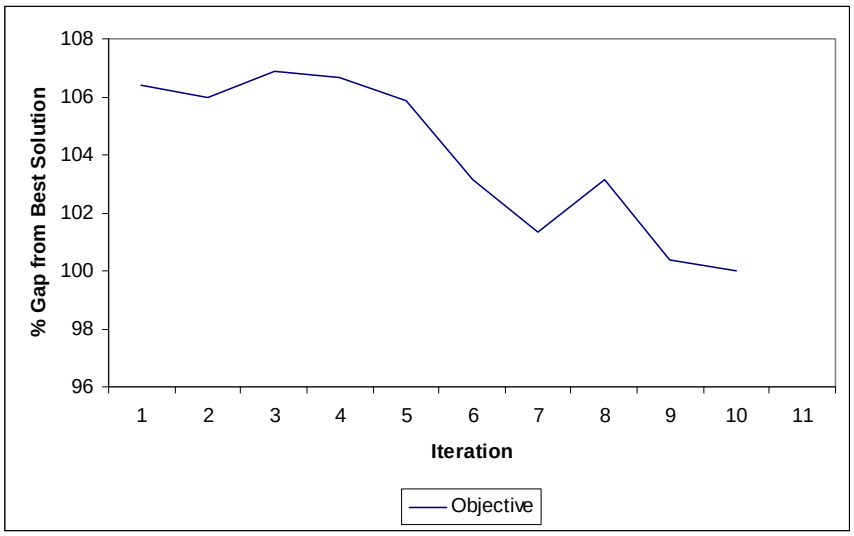

Figure 9: Objective Improvements for Market M2

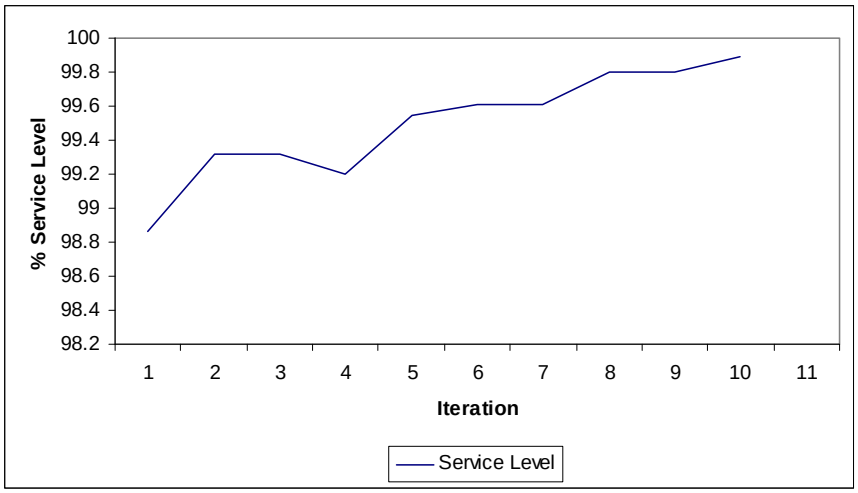

Figure 10: Service Level Improvements for Market M2

Since this is a tactical problem solved once at the beginning of a season, the running times of 24 hours are acceptable. Nevertheless, shorter times would be desirable from the what-if analysis point of view.

7. Concluding Remarks

The discussed PM operations problem is of vital importance to express package carriers. It provides a tremendous cost savings opportunity. Even a small relative improvement leads to substantial savings since the savings are assessed every day and for each market. Due to the underlying complexities and the sheer size, it is a tough nut to crack. Standard approaches such as integer programming do not yield satisfactory solutions even after relaxing several business rules. On the other hand, manual solutions provided by experienced engineers leave savings on the table. The proposed state-of-the-art dynamic programming algorithm provides several benefits. First, it establishes substantial savings. At the same time, the service level as measured by the number of on-time delivered packages is also improved. Finally, a successful deployment of an information system would ease the work load of engineers.

Due to the inherent complexities, modeling is a tedious task that must not be taken lightly. The notion of attributes and the attribute vector ease the notational burden by simplifying the state space and actions. The underlying dynamic programming algorithm uses a linear value function approximation, which is updated by using gradient information.

On the operational side, we model the option of vehicles making intermediate stops to pick up additional containers and/or packages. This is a non-trivial extension requiring careful modeling. Stations mostly relying on manual sorting allow the workers to prioritize sorting by manually picking the desired packages. Modern automated stations allow the packages to be sorted proportionally based on their arrival mixture. The latter, which is a focus of this work, creates nonlinear relationships in system dynamics. They further complicate the algorithmic development.

The presented model and algorithm are moving from the proof-of-concept phase to the production setting. To this end, the company is currently implementing a production version of the algorithm.

Acknowledgements

The authors would like to thank FedEx Express for providing the motivation for this work and the data used in computational experiments. Special acknowledgements are directed towards Bryan D'Souza, Mike West, and Faisal Zaiman.

We would also like to thank Professor Warren Powell for his helpful suggestions regarding modeling and notation.

Bibliography

Armacost, A., Barnhart, C., Ware, K., (2002). Composite Variable Formulations for Express Shipment Service Network Design. *Transportation Science*, **36**, 1-20.

Bertsekas, D., Nedic, A., Ozdaglar, A. (2003). *Convex Analysis and Optimization*. Athena Scientific, Belmont, MA, USA.

Crainic, T. G. (2003). Long-haul Freight Transportation. In *Handbook of Transportation Science* (R. W. Hall, editor). Kluwer Academic Publishers, Boston, MA, USA.

Crainic, T. G., Kim, K. H. (2007). Intermodal Transportation. In *Handbook in Operations Research and Management Science: Transportation* (C. Barnhart and G. Laporte, eds). Elsevier, Amsterdam, The Netherlands.

Kim, D., Barnhart, C., Ware, K., Reinhardt, G. (1999). Multimodal Express Package Delivery: A Service Network Design Application. *Transportation Science*, **33**, 391-407.

Powell, W., Carvalho, T. (1998). Dynamic Control of Logistics Queueing Networks for Large-Scale Fleet Management. *Transportation Science*, **32**, 90-109.

Powell, W.B., Shapiro, J. and Simao, H.P. (2001). A Representational Paradigm for Dynamic Resource Transformation Problems. In *Annals of Operations Research on Modeling* (C. Coullard, R. Fourer, and J. H. Owen, eds).

Powell, W. B., Topaloglu, H. (2003). Stochastic Programming in Transportation and Logistics, In *Handbooks in Operations Research and Management Science: Stochastic Programming* (A. Shapiro, A. Ruszczynski, eds.). Elsevier, Amsterdam, The Netherlands.

Powell, W. B., Bouzaiene-Ayari, B., Simao, H.P. (2007). Dynamic Models for Freight Transportation. In *Handbook in Operations Research and Management Science: Transportation* (C. Barnhart and G. Laporte, eds). Elsevier, Amsterdam, The Netherlands.

Schenk, L. (2005). *Intra Market Optimization for Express Package Carriers*. Master's Thesis, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign. Urbana, IL, USA.

Schenk, L., Klabjan, D. (2008). Intra Market Optimization for Express Package Carriers. (To appear in *Transportation Science.*)