Decision Models for the Knapsack Problem with Stochastic Returns and Capacity Requirements in Air Cargo Revenue Management

312904 Casper Rademaker

A thesis submitted for the degree of Master of Science in Econometrics and Management Science, Erasmus School of Economics, Erasmus University Rotterdam, The Netherlands.

November 2013

Academic supervisor: dr. Adriana Gabor
Co-reader: dr. Wilco van den Heuvel

Abstract

In this thesis, the effectiveness of various proposed strategies for the accept/reject decision regarding the knapsack problem with items with stochastic returns and capacity requirements are analyzed. The knapsack problem is a problem with a list of items with each its own return and capacity requirements. Such kinds of knapsack problems are common in air cargo revenue management in the booking process for free-sale cargo. The goal is to maximize returns while not exceeding knapsack capacities. The most common strategy is to accept requests whenever their return per capacity unit is at least equal to some threshold value. It is shown that even when setting these threshold values optimally, more complex decision strategies exist that perform better. These strategies include making the threshold value variable over the booking period, dividing the available capacity over several buckets with each its own threshold value and a strategy combining both. The results show that the last strategy combines the benefits of the two individual strategies and significantly outperforms them. However, determining the optimal decision variables for this strategy was found to be a very difficult problem in practice. A dynamic programming algorithm for the accept/reject decision is also proposed which performs better than the best found heuristic solutions in any other strategy in multidimensional cases.

Keywords: bid-prices, air cargo, revenue management, stochastic knapsack

TABLE OF CONTENTS

LIST OF TABLES, FIGURES, EQUATIONS, MODELS AND ALGORITHMS

CHAPTER 1 INTRODUCTION

This thesis is related to the field of revenue management in air cargo. The thesis is inspired by current challenges faced by the Revenue Management department of Air France-KLM-Martinair (AF-KL) Cargo, world's largest non-integrator cargo airline.

Revenue management is the practice of determining what is the lowest acceptable price to sell a unit of capacity space for a certain good, in order to maximize total returns. In air cargo, the capacity under consideration is cargo space on airplanes in three dimensions: weight, volume and pallet positions.

Passenger airlines are known to be the birth ground of revenue management. The advances made in revenue management in the passenger field can however not be directly translated to cargo revenue management. For instance, whereas passengers simply fill up a type of seat, such as a seat of economy class or business class, cargo is a more complex matter where one type of cargo fills capacity in three dimensions, of which weight and volume are continuous. Each cargo to be transported has its own specifications of these dimensions, thus cargo revenue management models have to be much more intricate to be able to deal with the much larger diversity of possible cargo specifications. However, due to the increased complexity, one might also expect that revenue management practices might have even larger merit in the cargo context than it has had in the passenger context.

Various problems in the field have been formulated in the literature, including overbooking, the balance between long-term contracts and spot deals and different routing options between locations (for an overview, see Kasilingam, 1996). The central problem in cargo revenue management is the booking problem, which is the problem of deciding whether to accept or reject a current request with certain known returns and capacity requirements, given a certain capacity on a route and given unknown future further demand on that route. Booking requests come in one by one, and, in the pure problem, it is assumed that it is not possible to cancel requests that have been acepted earlier in time. In scientific literature, this is a multidimensional knapsack problem with stochastic demand.

The central research question in this thesis is whether the current implemented strategies common in air cargo revenue management can be improved by using more advanced strategies and how large the improvement would be. The strategies analyzed can easily be applied in settings other than air cargo as well.

For an overview of early implementations of revenue management at KLM Cargo, pre Air

France-merger.see Slager and Kapteijns (2003). Particularly, it should be noted that AF-KL has already implemented cargo revenue management procedures, namely static bid-price models with changing bid-prices troughout the booking period due to recourse taking into account remaining capacities, that significantly improve expected revenues over all accepted bookings, compared to a first-come-first-served (FCFS) strategy. The strength of the implemented strategy is that it is quite parsimonous, however, because of its relative simplicity it is likely that other strategies will perform better. Evaluating the benefit of more complex strategies is the central topic of this thesis.

In Chapter 2 the problem and research question that are central to this thesis will be described. In Chapter 3, a literature review will be conducted regarding other papers that have been written on the subject. Also, it will be discussed what is novel in this research compared to earlier research on the subject. In Chapter 4, various possible bid-price strategies are discussed in words and in Chapter 5 the mathematical models are formalized with regards to the decision problem of determining the optimal decision variables in these bid-price strategies. In Chapter 6 a dynamic programming algorithm will be described that serves as an alternative strategy to bid-price strategies for the booking problem. In Chapter 7 the real-life data to be used in an investigation into the performance of the various strategies for the booking problem will be described in detail. In Chapters 8 and 9 the performance of the bid-price strategies will be discussed. In Chapter 8 the analysis is based on small instances that could be solved to close-to optimality. In Chapter 9 the analysis is based on very large instances that could not be solved to optimality and decision variables were chosen based on heuristics. The first analysis is more relevant for determining the improvement potential moving from certain strategies to others, while the second analysis is more relevant for practice: namely how much of this improvement potential is likely to be fulfilled in practice. In Chapter 10 the dynamic programming algorithm's performance compared to the optimized bid-price strategies on small instances will be analyzed. In Chapter 11 the dynamic programming algorith's performance on large instances compared to the bid-price strategies in combination with heuristics will be analyzed.

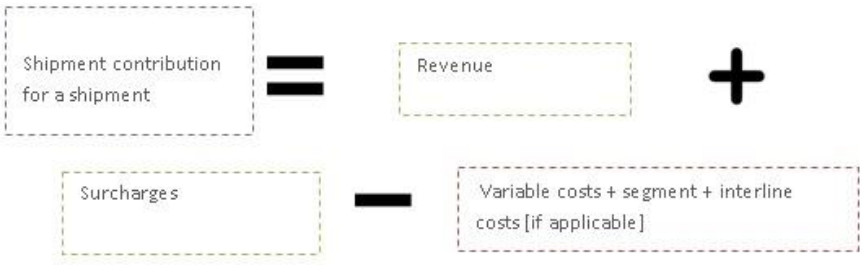
Readers who are not comfortable in the area of Operations Research may skim Chapter 3 and skip Chapters 5 and 6 without loss of clarity of argument.

Chapter 2 Problem Definition and Research Question

The business context of this thesis is cargo airlines. Cargo airlines transport cargo from point A to point B. Their direct customers are called forwarders, which are specialized logistic organizations acting as middle-man in the relationship between end-customers, for instance shops or producers, and airlines. A combination of a specification of cargo and an itinerary, which is the exact travel route from point A to B including transshipment points and modes of transport, is called a shipment. It should be noted that not all segments in an itinerary have to be flights; in fact, almost all intra-European cargo travel by AF-KL is transported by trucks. Also, not all flights have to be executed by the shipping airline, some are operated by another airline, in which the shipping airline pays this operating airline an interline fee.

Current tendering procedure:

Central to the problem definition of this thesis is the current procedure of accepting offers made by forwarders, the customers of cargo airlines, for a shipment. The main criterion on which the acceptation decision is based, is shipment contribution (SCb) per kg or m^3 for that shipment, which is a cargo airline's specification of contribution margin. Contribution margin equals revenue per unit minus variable costs per unit, where a unit is defined as a kilogram in weight or a cube in volume of cargo. An airplane's cargo capacity is bounded by weight as well as volume, so the appropriate unit to use depends on which of these two is the stricter constraining factor. As mentioned before, an airplane's cargo capacity is also bounded by it's amount of pallets positions available, but this is not often constraining. In AF-KL's experience, in the great majority of all flights, volume is the stricter constraining factor, in which case SCb per m^3 is the appropriate metric to use. The exact buildup of total SCb per shipment is as follows:



Equation 1

In this equation, revenue consists of the base-rate per chargeable kg multiplied by chargeable weight in kg. The distinction between actual weight and chargeable weight is

discussed later in this section. Surcharges consist of a fuel surcharge and a security surcharge per actual kilogram of weight. Variable costs consist of handling costs at the stations, segment costs consist of actual fuel costs for the flight and costs of trucking if applicable.

Fixed costs are deliberately not a part of the shipment contribution's definition. The great majority of cargo is transported on flights that also carry passengers. Thus, the decision to fly these flights has already been made and fixed costs are thus regarded as sunk costs. In the remainder of this thesis, flights will be considered to be shared passenger-cargo flights. Dividing SCb by the total volume in m^3 or weight in kg of a shipment yields SCb per m^3 or SCb per kg for this shipment. Obviously, the larger SCb per unit is, the more attractive a certain quote from a forwarder is for an airline, holding other variables constant. Airlines typically set a threshold value for SCb per unit for a shipment, called an Entry Condition (EC) in AF-KL, or in the broader academic literature, a bid-price. If the SCb per unit for a request on a shipment is at least equal to the EC for that shipment, that request is in principle accepted and otherwise it is in principle rejected. Price deducted by EC is dubbed "additional SCb".

When capacity exceeds forecasted demand, the EC is usually set at a level that revenues are able to cover variable costs plus a safety margin, dubbed 'rock-bottom' EC by AF-KL. For higher demand-flights, the EC is set at a higher level, such that the customers with the lowest willingness to pay can be rejected, yet the flight will still be full enough.

It is possible and usual to create multiple entry conditions for a flight, inducing price differentiation. For instance, half of the flight capacity might have an entry condition A, and the other half might have an entry condition B. This strategy is yet to be fully implemented by AF-KL and investigating the benefits of such a strategy compared to the regular strategy is one of the research topics in this thesis.

Considerations regarding volume and weight:

As noted before, the primary constraint in an airplane's cargo capacity is its volume, but basic cargo rates are based on weight. This would mean, without further sophistication, that dense goods, meaning goods with a large weight to volume ratio, would be priced the same as low density goods with the same weight, while in most cases, namely when the flight involved is volume-constrained, the first case would be preferable for the airline. For this reason, the concept of chargeable weight is introduced by IATA, the international aviation regulatory association. The industry standard density of goods is defined as one ton (1000 kg) per 6 m^3. Any goods having lower density than this standard are defined as being low

density goods, the remaining goods are defined as high density goods. For high density goods, chargeable weight is defined as being equal to actual weight. For low density goods, chargeable weight in tons = actual volume in $m^3/6$, note that under the industry standard, a 1 m^3 piece would weigh $1/6^{th}$ of a ton. This practically means that the normally relatively unattractive low density goods, compared to high density goods or the industry standard, have their effective rates increased. They are now priced as if their density was one ton per 6 m^3. However, high density goods do not get a similar discount. Note that the effect of this is that the high density goods still have relatively attractive rates, from the airline's side, given that a shipment is volume-constrained rather than weight-constrained.

In the remainder of this thesis, weight will refer to actual weight unless stated otherwise, and SCb's related to the respective requests will be adjusted in the way their density requires.

There is also a definitional distinction to be made regarding volume. Volume can be defined in two ways: actual physical volume or 'water volume' and operational volume. Goods have a certain physical volume, but might also have restrictions regarding safe transportation. For instance, on some goods, other goods may not be stacked. The volume that could otherwise be used but because of this kind of restriction cannot be utilized is added to the physical volume to make up operational volume. Operational volume is always larger than or equal to physical volume. Unless stated otherwise, volume will be assumed to refer to operational volume, because this is the volume that is by far most important in practice, for instance, when considering volume capacity on a flight, it is total operational volume that has to smaller than the volume capacity of the airplane.
A flight that is (very close to) full on one of the capacities is called a constrained flight.

Research question:

The central research question in this thesis is the following:

How well does a strategy of using a single EC or bid-price perform in the booking problem compared to the a posteriori optimum, in terms of expected cumulative returns obtained? Is it possible to increase expected cumulative returns by implementing more advanced strategies for the booking problem?

See Appendix B for an overview of all terms introduced in this chapter.

In this chapter, the business context surrounding the booking problem in air cargo has been described. Also, the research question was introduced. In the next chapter, a literature review will be discussed in which various results related to methods for the booking problem in air cargo are investigated.

Chapter 3 Literature Review

In the previous chapter, the problem definition and research question have been formulated. It has been discussed that the central problem in this thesis is the booking problem in air cargo. In this chapter, a literature review will be conducted regarding academic studies focusing on the same problem.

Disregarding any aspects related to the upfront unknown, stochastic demand occurrences, the booking problem can be seen as a multidimensional knapsack problem. A multidimensional knapsack problem is one in which certain items are to be selected, while the knapsack has maximum capacities in various dimensions (Frieze & Clarke, 1984). Each item has a certain utility value and the goal is to maximize the sum of utility values of the items that are selected, such that the capacities of the knapsack are not exceeded. In the cargo case, the capacity dimensions include weight, volume and available pallet positions, although the last is in practice very unrestrictive and is often disregarded from models (Pak & Dekker, 2004). The knapsack problem is an NP-hard problem and for large instances, it might take an extremely large time to solve the problem to optimality.

Polynomial approximation algorithms exist that asymptotically approach optimality when the number of items in the problem grow to infinity (cf. Rinnooy Kan et al., 1993). However, this still leaves the problem of stochastic demand occurences unanswered. This problem is partially solved by Papavastrou et. al. (1996) who developed a dynamic programming approach that solves the accept/reject decision for potential knapsack items, where all knapsack items arrive in different points of time, each has a stochastic weight and utility and rejected knapsack items cannot be recalled at a later point in time. Thus, instead of maximizing returns as in the deterministic problem, their approach maximizes expected utility or returns, as is the goal in this thesis as well. However, they only consider the one-dimensional knapsack problem and their results are dependent on some assumed distributions for capacity requirements and utility, namely, that the conditional probability density functions for the capacity requirements given returns are concave. Thus, their method is not general and this requirement may not be met in real life.

Another related problem is determining the optimal overbooking level. Kasilingam (1996) developed an integral equation that can be solved for the optimal overbooking level given the density functions for the show-up rate and cargo capacities, the last of which are stochastic on combined passenger-cargo flights due to the fact that passenger luggage has a

higher boarding priority than business cargo, and given deterministic offloading costs. Depending on the complexity of the density functions used, the equations can either be solved to optimality analytically or be optimized using numerical methods.

Various practical approaches to optimizing the solution to booking problems have been developed in the academic literature. Various papers have used a bid-price approach, in which the profit rate of accepting a certain cargo offer is known, but the optimal threshold level of the bid-price is to be determined, i.e. it is not obvious what the lowest bid is that the airline should accept for this cargo. By using a bid-price approach, it is prevented that early cargo with a relatively low return is filling up space that could be filled up by later cargo with relatively higher returns. Thus the threshold bid-price per kg or m^3 is set to the levels that optimize expected total returns, while cargo return rates, future demand distributions and their volume and weight distributions are considered to be known. This corresponds to our situation of the SCb for a request being known but the threshold price for accepting the request is not. The threshold bid-price can also be interpreted as the value for which a unit of capacity can be expected to be sold in the future at a certain point of time (Pak & Dekker, 2004).

The first paper addressing the booking problem in air cargo is by Pak and Dekker (2004). In their model, a bid is accepted if the profit rate associated with the bid is higher than the opportunity cost rate associated with the offer, i.e. the expected future revenue lost by accepting the bid, because less weight and volume will be available. Obviously, in order for this condition to work, some kind of expected revenue function that can evaluate the opportunity costs involved with accepting the bid needs to be specified. Given this revenue function and given that all demand occurrences, including their returns and capacity requirements, are known in forehand, Pak and Dekker have developed both a MIP-model and a greedy algorithm that optimize the bid-prices, but the assumption of clairvoyance is obviously a bit utopian. Their more general approach is to use simulation to generate the future demand occurrences. Over each simulation run and demand instance, optimal bid-prices per weight and volume rate are determined, using a polynomial time greedy heuristic for solving multi-dimensional knapsack problems, described in Rinnooy Kan et al. (1993). By averaging over all demand instances, the bid-prices that optimize average profit over all instances can be determined.

Pak and Dekker study the performance of their algorithm and conclude that it leads to

almost 10 percent profit increases compared to the most obvious and simplest solution of using the shadow prices for the weight and volume constraints obtained from the LP-relaxation of the MIP. Interestingly though, they also conclude that the use of the greedy algorithm described earlier, combined with simulated demand occurrences, lead to even slightly better results, but that the use of it was still too computation intensive to be used in online decision making at the time.

Han et al. (2010) model practically the same problem using a Markov chain. The states of the model keep track of the cumulative weight and volume booked at a certain moment. Cargo offers are accepted if the profit rate is higher than the threshold bid-price per kg multiplied by chargeable weight. Using a returns function that links the expected number of times various states are visited with the expected revenue resulting from visiting these states, they developed a closed-form expression for total expected revenue as a function of the threshold bid-prices. The optimal bid-prices can then be determined using some unconstrained optimization method, as the revenue surface is unimodal. Their simulation efforts show that their method results in higher overall revenue than Pak and Dekker's approach in most situations, although the magnitude of the difference is very dependent on the parameterization.

Huang and Hsu (2005) focus specifically on the risks associated with overbooking and stochastic cargo capacities: namely the fact that cargo may need to be offloaded when the real capacity is smaller than the booked cargo. They developed a dynamic programming algorithm that optimizes overall revenue given profit rates and demand probabilities in each period by optimizing the accept/reject decision per booking request. In an empirical study they also come to an interesting conclusion regarding no-show penalties: total expected revenue is much more sensitive to supply uncertainty than to no-show penalties, and they conclude that airlines might increase their expected revenue by increasing penalties, although they did not investigate the effect of this policy on the airline's attractiveness to customers. Their approach might be considered to be too computationally demanding to use in a continuous online accept/reject decision cycle.

Amaruchkul et. al. (2007) also use a dynamic programming approach policy for optimizing the accept/reject decision for booking offers. An advantage of their approach is that they consider actual volume and weight of cargo, i.e. the volume and weight of the cargo that are actually delivered by the customers to the airline, to be different from requested volume and weight, a distinction that is important in practice but is not modeled in the models

described before. Interestingly, they come up with a decision heuristic using bounds on some statistic relationships that dramatically reduces the computational requirements associated with deriving the optimal policy.

As can be understood from the above overview, even if an appropriate model exists for the deterministic booking problem, the issue of stochastic demand still needs to be addressed. Various difficulties are in place here that are unique to the cargo segment and have been discussed in the literature, including the fact that the booking period for free sales is quite short, it opens 14 days before booking. Another difficulty is the fact that the number of bookings is most of the times very small, but with very volatile weight and volume requirements (Kasilingam, 1996). Another, more practical, difficulty is due to the fact that residual demand prediction requires information on all booking requests made, including those booking requests that are rejected (Moussawi and Çakanyildirim, 2005). However, in general rejected booking requests are not kept track of in most airline's databases. This is currently also true in the case of AF-KL.

The following table summarizes the various papers that have been discussed on methods for improving performances of strategies for the air cargo booking problem:

Table 1: Overview of methods

Authors (year)	Method used	Decision variables
Papavastrou et. al. (1996)	Dynamic programming, deadlined knapsack problem with stochastic weights and returns	Accepting/rejecting incoming items
Pak and Dekker (2004)	Simulation, bid-price policy	Bid-prices for weight and volume, knowing which fixes the accept/reject decision
Rinnooy Kan et al. (1993	Greedy algorithm, multidimensional knapsack problem	Accepting/rejecting incoming items
Han et al. (2010)	Markov chain analysis, bid-price policy	Bid-prices for weight and volume
Huang and Hsu (2005)	Dynamic programming, overbooking	Accept/reject decision, no-show penalty, overbooking rate
Amaruchkul et. al. (2007)	Heuristics based on dynamic programming	Accept/reject decision

In this chapter a literature review has been conducted regarding methods that have been developed for the booking problem in air cargo. As can be seen, there has been quite some research done into determining optimal bid-prices. However, what is missing is an analysis in the exact merits of a bid-price based strategy for the booking problem with stochastic entries. Assume it is possible to determine bid-prices optimally, then there is still the question of whether the strategy of accepting a request whenever its return per volume unit

is at least equal to the bid-price is an optimal strategy. In the next chapter, the various strategies that will be tested in an empirical study will be described.

CHAPTER 4 BID-PRICE STRATEGIES

As found in the business context section, a bid-price policy for individual flights has already been implemented in the air cargo industry. The academic literature is focused on determining the optimal values of these bid-prices. However, there is little discussion on how optimal a static bid-price strategy is. This thesis will focus on determining how well other strategies than a static bid-price strategy perform, both in how large the gap in expected returns is to the optimal a posteriori knapsack solution and in ease of determining well-performing decision variables in practice.

Various strategies exist or have been proposed in AF-KL for the accept/reject decision. These strategies will first be discussed briefly. In later chapters of this paper, their respective effectivities will be analyzed based on real-life data.

Strategy 1a:

This strategy uses a single entry condition (EC) based on volume, that is independent of time. I.e. the entry condition remains constant throughout the booking period. A booking request is accepted if its ratio $\frac{SCb}{volume}$ is larger than or equal to the EC. It is clear that there exists an optimal EC within this strategy, however, this does not mean that there do not exist other strategies that perform better. This strategy is expected to work reasonably well for volume-constrained flights, but it only takes request differentiation into account very roughly. It is expected to work well for very 'smooth' booking requests' SCb/m^3 distributions or when the various distributions of booking requests are independent of time, i.e. the profile of incoming requests is about the same at the start of the booking period as it is at any other moment in the booking period.

Strategy 1b:

This strategy uses a time-dependent EC based on volume. That is, the EC is allowed to be changed after every booking request, or after an interval of any other number of requests. A booking request is accepted if its ratio $\frac{SCb}{volume}$ is larger than or equal to the specific EC applicable at that period. As in strategy 1a, the goal is to find the set of EC's that maximizes expected cumulative SCb. This strategy is expected to work much better than strategy 1a in

cases where the booking requests' distributions of returns and/or capacity requirements differ significantly over time.

Strategy 2a:

Dividing the volume capacity of an airplane into various (*B)* buckets, and setting time-independent EC's on each of these buckets. A booking request will be filling the capacity of the highest bucket such that the booking request's ratio $\frac{SCb}{volume}$ is still at least equal to that bucket's EC and such that there is still capacity left in that bucket. Eventual residual volume left after using this rule will be filling capacity respecting the same rule with updated remaining bucket capacities. A booking request is still rejected if its volume is larger than the sum of remaining volumes over all buckets for which that bucket's EC is smaller or equal to that request's $\frac{SCb}{volume}$, i.e. it is not possible to accept partial requests.
For example: consider a flight with 900 volume, divided in three buckets of 300 each. The EC's are set as follows: 400, 500 and 600. Consider the first incoming request having 350 volume and 550 SCb/m^3. It will not fill the third bucket as its SCb per m^3 is not high enough. It will fill the second bucket entirely. The 50 residual volume will be filling the first bucket, which now has 250 volume remaining. Consider the second incoming request also having 550 SCb/m^3 and 350 volume. It will not be accepted because the third bucket has too high EC for this request and the first and second buckets only have a sum of 250 remaining volume. This strategy is expected to perform well in the cases of well differentiated booking requests, e.g. there are several peaks in the distribution of SCb/m^3 per booking request.

Strategy 2b:

In this strategy, we combine strategies 1b and 2a, namely we incorporate buckets with ECs that are time-dependent. The conceptualized increases in expected cumulative SCb of this strategy vis-à-vis strategy 1a are equal to the sum of the increases of strategies 1b and 2a, although the total benefit will probably be smaller than the sum of the two individual increases, due to the fact that the benefits of the individual strategies probably have some overlap.

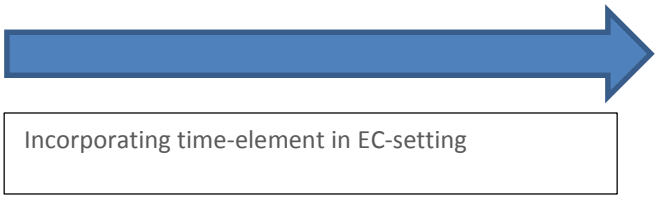
Note that in the thesis, in general, for all strategies an EC based on volume will always be used, even if in some cases the capacities and demand are set such that the flight is likely to be weight-constrained. Besides parsimony, another reason for this is that if more sophisticated strategies can close the optimality gap with the a posteriori optimal knapsack

solution sufficiently, then the whole process of deciding dual bid-prices, one for volume and one for weight, can be neglected.

In some sections, the effect of having dual bid-prices compared to a single bid-price will be analyzed. Whenever one of the strategies is executed with dual bid-prices, this will be explicitly mentioned.

Framework of strategies:

Strategies 1a to 2b can be summarized in the following framework.



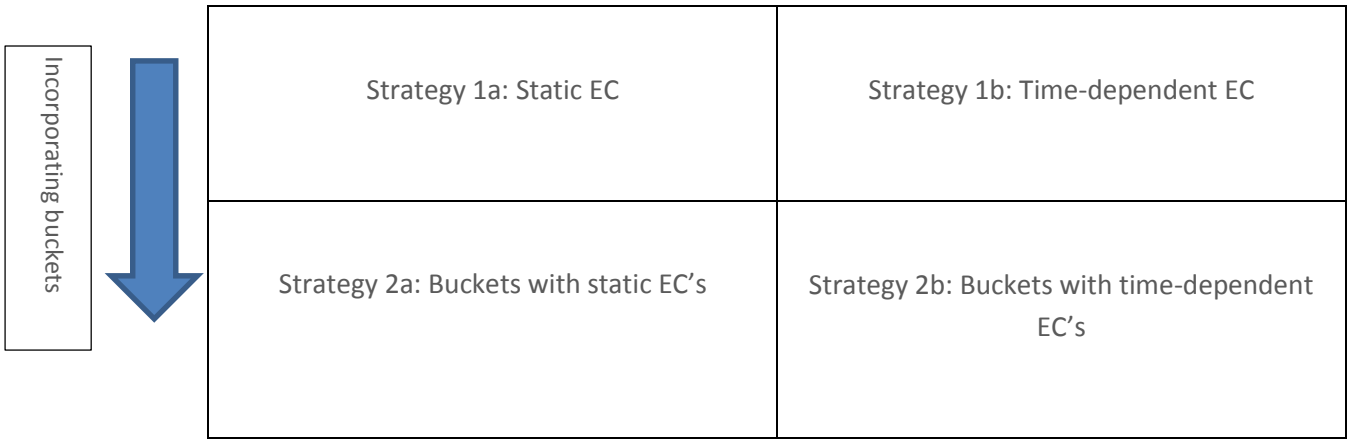


Figure 1: Strategies to be analyzed

In this chapter, various bid-price strategies for determining which items to reject and which to accept in the booking problem in air cargo have been discussed. In the next chapter, these strategies will be formalized in mathematical models.

CHAPTER 5 MATHEMATICAL APPROACHES FOR BID-PRICE STRATEGIES

In this chapter, mathematical approaches for determining the optimal decision variables in bid-price strategies will be formalized. Note that the strategies discussed here assume a two-dimensional problem in weight and volume, although it is conceptually not difficult to extend the models to include more dimensions.

Strategy 1a:

Strategy 1a is a problem in one variable, namely the static Entry Condition. This means it is not difficult to find the optimal Entry Condition enumerating over all values of SCb/m^3 for a certain list of request. When EC's have to be acquired that are robust over varying demand instances, we should repeat this procedure a large number of times using simulation, by generating demand instances from appropriate distributions.
This simulation should have as input the distribution of incoming requests, the volume and weight capacity of a flight, the number of requests to be generated and the entry condition used. The simulation then generates for each run the desired number of requests from appropriate distributions, namely the distributions of a request's volume, weight and SCb and shows whether each request is accepted or rejected, the remaining weight and volume after each request and the cumulative SCb at each time period.
The main use of this simulation is to calculate the average cumulative SCb over a number of simulation runs for a certain request distribution and a certain EC-setting.
An incoming request is accepted when both of the following conditions are met and rejected otherwise:
1) The request's SCb/volume unit is at least equal to the EC at that period.
2) The request's volume requirement, respectively weight requirement are smaller or equal to the remaining volume, respectively remaining weight at that period.

The simulation is not complex and has fast running times, namely approximately 0.3 seconds for 100 requests and 100 simulation runs[1]. This implies that in the case of strategy 1a it can also easily be used to determine an approximate optimal EC by iterating over all reasonable

[1] When solving times are mentioned, the used hardware is an HP EliteBook 2530p with Intel Core 2 Duo CPU (2 cores), 4GB RAM, Windows 7 64-bit. AIMMS 3.13 SU1 was used for modeling MIPs and unless specified otherwise, the commercial solver GUROBI 5.5 was used for solving the MIPs.

ECs, by starting with an EC of 0 and increasing the EC to be tested by a certain increment after each iteration, until some kind of stopping criterion is reached and a posteriori taking the EC with the highest average cumulative SCb. For instance, the safe stopping criterion choice is when the EC to be tested is larger than the maximal value of SCb/volume unit in the distribution.

MIP strategy 1b:

The mathematical program for determining the a posteriori optimal EC under strategy 1b for a certain list of booking requests is as follows. Note: in this MIP and all following MIPs, round down found EC's to the next significant digit. The objective value of the MIP is correct but the MIP might set the EC's slightly too high (only in the last digit).

Index:

t: for period. Each period a booking request comes in

Variables:

y_t^1: binary indicator variable, equal to 1 if EC smaller than or equal to SCb/m^3, 0 otherwise

y_t^2: binary indicator variable, equal to 1 if the request at time *t* still fits weight-wise, 0 otherwise

y_t^3: binary indicator variable, equal to 1 if the request at time *t* still fits volume-wise, 0 otherwise

x_t: binary decision variable, equal to 1 if the request at time *t* should be accepted, 0 otherwise

EC_t: entry condition for period *t*

Parameters:

s_t: total SCb of the request in period *t*

w_t: weight of the request in period *t*

v_t: volume of the request in period *t*

$q_t = \frac{s_t}{v_t}$: SCb/volume unit in period *t*

$weightcapacity$: weight capacity of the flight

$volumecapacity$: volume capacity of the flight

ε: smallest positive number possible

$$\max \sum_{t=1}^{T} x_t s_t \qquad s.t. \qquad [1]$$

$$y_t^1 \geq \frac{1}{\max(q_t)+1}(q_t - EC_t) + \varepsilon \quad for\ t = 1, \dots, T \quad [2]$$

$$EC_t - q_t \leq \max(q_t)(1 - y_t^1) \quad for\ t = 1, \dots, T \ [3]$$

$$y_t^1 \in \{0,1\}\ for\ t = 1, \dots, T \ [4]$$

$$w_t - (weightcapacity - \sum_{j=1}^{t-1} x_j w_j) \leq \max(w_t)(1 - y_t^2)\ for\ t = 1, \dots, T \ [5]$$

$$y_t^2 \in \{0,1\}\ for\ t = 1, \dots, T \ [6]$$

$$y_t^2 \geq \frac{1}{weightcapacity + 1}\left((weightcapacity - \sum_{j=1}^{t-1} x_j w_j) - w_t\right) + \varepsilon \ [7]$$

$$v_t - (volumecapacity - \sum_{j=1}^{t-1} x_j v_j) \leq \max(v_t)(1 - y_t^3)\ for\ t = 1, \dots, T \ [8]$$

$$y_t^3 \in \{0,1\}\ for\ t = 1, \dots, T \ [9]$$

$$y_t^3 \geq \frac{1}{volumecapacity + 1}\left((volumecapacity - \sum_{j=1}^{t-1} x_j v_j) - v_t\right) + \varepsilon\ for\ t = 1, \dots, T \ [10]$$

$$x_t \geq y_t^1 + y_t^2 + y_t^3 - 2 \quad for\ t = 1, \dots, T \ [11]$$

$$x_t \leq \frac{1}{3}(y_t^1 + y_t^2 + y_t^3) \quad for\ t = 1, \dots, T \ [12]$$

$$x_t \in \{0,1\} \quad for\ t = 1, \dots, T \ [13]$$

$$EC_t \geq 0\ for\ t = 1, \dots, T \ [14]$$

MP 1: MIP for strategy 1b, one scenario

The objective function maximizes the sum of SCb's over all accepted requests. Constraints 11 to 13 guarantee that a request is accepted when all three conditions are satisfied and a request is rejected when less than three of the conditions are satisfied. Constraints 2 to 4 set the binary variable related to the first condition for a request to 1 whenever the EC is smaller or equal to the request's SCb/m^3, 0 otherwise. Constraints 5 to 7 set the binary variable related to the second condition for a request to 0 whenever the request does not fit weight-wise, 1 otherwise. Constraints 8 to 10 are equivalent constraints for the volume dimension.

Note that this model optimizes the entry conditions purely based on one scenario, i.e. a list of *T* request. To determine the entry condition that is robust over various scenarios the above model should and can easily be extended in a stochastic programming manner. All variables and parameters, except EC and the weight and volume capacities which are assumed to be constant over all scenarios, should simply be augmented with a subscript *sc*

for scenario. As for the objective function, the average sum of all SCb's over all accepted requests over all scenarios should be taken. This results in the following model. Here, pn_{sc} equals the normalized probability of scenario *sc*, $pn_{sc} = \frac{p_{sc}}{\sum_{sc=1}^{N} p_{sc}}$, where $p_{sc}$is the marginal probability of scenario *sc*.

Note: the sum of marginal probabilities of the used scenarios is usually smaller than 1 due to the fact that there are more scenarios possible than the ones consider. In fact these marginal probabilities might even be smaller than the machine epsilon, an upper bound on the relative error due to rounding in floating point arithmetic, and thus may not be able to be determined accurately using conventional methods. In the succeeding analyses in this thesis, pn_{sc} is set to uniform probability 1/N for all *sc*.

$$\max \sum_{sc=1}^{N} \sum_{t=1}^{T} pn_{sc} x_{t,sc} s_{t,sc} \qquad s.t. \qquad [1]$$

$$y_{t,sc}^{1} \geq \frac{1}{\max(q_{t,sc}) + 1}\left(q_{t,sc} - EC_t\right) + \varepsilon \quad for\ t = 1, \dots, T, sc = 1, \dots, N \quad [2]$$

$$EC_t - q_{t,sc} \leq \max(q_{t,sc})\left(1 - y_{t,sc}^{1}\right) \quad for\ t = 1, \dots, T, sc = 1, \dots, N \ [3]$$

$$y_{t,sc}^{1} \in \{0,1\}\ for\ t = 1, \dots, T, sc = 1, \dots, N \ [4]$$

$$w_{t,sc} - (weightcapacity - \sum_{j=1}^{t-1} x_j w_j) \leq \max(w_{t,sc})\left(1 - y_{t,sc}^{2}\right)\ for\ t = 1, \dots, T, sc = 1, \dots, N \ [5]$$

$$y_{t,sc}^{2} \in \{0,1\}\ for\ t = 1, \dots, T, sc = 1, \dots, N \ [6]$$

$$y_{t,sc}^{2} \geq \frac{1}{weightcapacity + 1}\left((weightcapacity - \sum_{j=1}^{t-1} x_j w_j) - w_{t,sc}\right) + \varepsilon \ for\ t = 1, \dots, T, sc = 1, \dots, N \ [7]$$

$$v_{t,sc} - (volumecapacity - \sum_{j=1}^{t-1} x_j v_j) \leq \max(v_{t,sc})\left(1 - y_{t,sc}^{3}\right)\ for\ t = 1, \dots, T, sc = 1, \dots, N \ [8]$$

$$y_{t,sc}^{3} \in \{0,1\}\ for\ t = 1, \dots, T, sc = 1, \dots, N \ [9]$$

$$y_{t,sc}^{3} \geq \frac{1}{volumecapacity + 1}\left((volumecapacity - \sum_{j=1}^{t-1} x_j v_j) - v_{t,sc}\right) + \varepsilon \ for\ t = 1, \dots, T, sc = 1, \dots, N \ [10]$$

$$x_{t,sc} \geq y_{t,sc}^{1} + y_{t,sc}^{2} + y_{t,sc}^{3} - 2 \ for\ t = 1, \dots, T, sc = 1, \dots, N \ [11]$$

$$x_{t,sc} \leq \frac{1}{3}\left(y_{t,sc}^{1} + y_{t,sc}^{2} + y_{t,sc}^{3}\right) \quad for\ t = 1, \dots, T, sc = 1, \dots, N \ [12]$$

$$x_{t,sc} \in \{0,1\} \quad for\ t = 1, \dots, T, sc = 1, \dots, N \ [13]$$

$$EC_t \geq 0 \ for\ t = 1, \dots, T \ [14]$$

MP 2: MIP for strategy 1b, multiple scenarios

The optimal solution for strategy 1b can never be worse off than the optimal solution for strategy 1a, as the EC's for each period could simply set to the optimal EC under strategy 1a, i.e. every period has the same EC.
Note that when using a single scenario, the optimal objective value will be equal to the solution of the equivalent multidimensional knapsack model. The knapsack solution can namely be translated to this MIP's solution by setting EC to infinity for items that should not be accepted and to zero for items that should be accepted.
When using multiple scenarios, this translation is not possible as the EC's for each *t* are valid over all scenarios.

Finally, note that the EC for strategy 1a can also easily be obtained using MP 2, by omitting the subscript *t* in EC, even though it is much more resource efficient to obtain a static EC by enumeration or simulation.

Considerations regarding solution's robustness over varying demand instances:

This MIP-formulation is quite complex and takes a rather long time to solve using commercial solvers for large *T* and *N*. For instance, using *T*=100 and *N*=15, it already takes about 10 minutes to solve this model with a 0.28% optimality gap using GUROBI 5.5. This results in EC's that are close to optimal for the 15 scenarios used. This is not likely to be a very robust solution as for each time period we only have considered a maximum of 15 distinct requests. Thus, the EC's are close to optimal for the considered scenarios but might perform quite bad for scenarios out of scope of the optimization. A suggested solution that is theoretically less well performing in the case of usage with a large number of scenarios but is likely to be much more robust in the case of a small number of scenarios considered is to divide *T* in *m* distinct sub-periods. Then, only the optimal EC's for the *m* sub-periods are determined. Thus, instead of *T* different EC's, $m < T$ EC's are determined. For period *t*, EC_r is used whenever $t \leq r \leq T$.

This approach has two concurrent and independent advantages:
1) Keeping available computation resources and time equal, more scenarios can be considered in the optimization, as the number of decision variables and constraints in the optimization is reduced.
2) Keeping the amount of scenarios equal, more requests are considered in determining each EC. For instance: using *N*=15 and *m* =1, $1 * 15 = 15$ requests are considered for

determining each EC. Using *N*=15 and *m* =5, $5 * 15 = 75$ requests are considered for determining each EC. Thus, the resulting EC's are likely to be much more robust for scenarios not considered in the optimization.

MIP strategy 2b:

The MIP for determining optimal bucket sizes and ECs is the following:

Index:
t: for period. Each period a booking request comes in
sc: for scenario
i: for bucket

Variables:
EC_{it}: entry condition for bucket *i*, period *t*
$ra_{t,sc}^{i}$: volume accepted in bucket *i*, period *t*, scenario *sc*
$y_{t,sc}^{1}$: binary variable, equal to 1 if there is still enough weight remaining for the request at time period *t*, in scenario *sc*, 0 otherwise
$ca_{t,sc}^{i}$: equal to remaining volume in bucket *i*, if $q_{t,sc} \geq$ EC, 0 otherwise, in period *t*, scenario *sc*
$y_{t,sc}^{2}$: binary variable, equal to 1 if the sum of remaining volumes in the buckets for which the SCb/volume unit is higher or equal to EC, in period *t*, scenario *sc*, 0 otherwise
$y_{t,sc}^{3i}$: binary variable, equal to 1 if the entry condition for bucket *i* is smaller than or equal to the SCb/volume for period *t*, scenario *sc*
$x_{t,sc}^{i}$: binary variable, equal to 1 if a certain positive amount of volume is accepted in bucket *i*, 0 otherwise, period *t*, scenario *sc*
$A_{t,sc}^{i}$: auxiliary variable, equal to min($v_{t,sc}$, $rv_{t-1,sc}^{i}$)
cn^{i}: bucket proportion (larger or equal to 0, smaller or equal to 1, sum up to 1 over all buckets)
$x_{t,sc}$: binary variable, equal to 1 if the request in period *t* should be accepted, 0 otherwise
$z_{t,sc}^{i}$is a binary auxiliary variable used to model an if-then type constraint.

Parameters:
B = number of buckets
pn_{sc} = normalized probability of scenario *sc*
$s_{t,sc}$: total SCb of the request in period *t*, scenario *sc*
$w_{t,sc}$: weight of the request in period *t*, scenario *sc*
$v_{t,sc}$: volume of the request in period *t*, scenario *sc*
$q_{t,sc} = \frac{s_t,sc}{v_t,sc}$: SCb/volume unit in period *t*, scenario *sc*
$weightcapacity$: weight capacity of the flight
$volumecapacity$: volume capacity of the flight
ε: smallest positive number possible

$$\max \sum_{sc=1}^{N}\sum_{t=1}^{T} pn_{sc}\, x_{t,sc} S_{t,sc} \qquad s.t. \qquad [1]$$

$$y^1_{t,sc} \geq \frac{1}{weightcapacity+1}\left(weightcapacity - \sum_{j=1}^{t-1} x_{j,sc} w_{j,sc}\right) + \varepsilon \;\; for\; t = 1,\dots,T, sc = 1,\dots,N \; [2]$$

$$w_{t,sc} - (weightcapacity - \sum_{j=1}^{t-1} x_{j,sc} w_{j,sc}) \leq \max(w_{t,sc})\left(1 - y^1_{t,sc}\right) for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [3]$$

$$y^2_{t,sc} \geq \frac{1}{volumecapacity+1}\left(\sum_{i=1}^{B} ca^i_{t,sc} - v_{t,sc}\right) + \varepsilon \;\; for\; t = 1,\dots,T, sc = 1,\dots,N \; [4]$$

$$v_{t,sc} - \sum_{i=1}^{B} ca^i_{t,sc} \leq \max(v_{t,sc})\left(1 - y^2_{t,sc}\right) for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [5]$$

$$y^{3i}_{t,sc} \geq \frac{1}{\max(q_{t,sc})+1}\left(q_{t,sc} - EC_{it}\right) + \varepsilon \;\; for\; i = 1,\dots,B, t = 1,\dots,T, sc = 1,\dots,N \; [6]$$

$$EC_{it} - q_{t,sc} \leq \max(q_{t,sc})(1 - y^{3i}_{t,sc}) \;\; for\; i = 1,\dots,B, t = 1,\dots,T, sc = 1,\dots,N \; [7]$$

$$x^i_{t,sc} \leq \frac{1}{3}\left(y^1_{t,sc} + y^2_{t,sc} + y^{3i}_{t,sc}\right) for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [8]$$

$$x^i_{t,sc} \geq \left(y^1_{t,sc} + y^2_{t,sc} + y^{3i}_{t,sc}\right) - 2 \; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [9]$$

$$A^i_{t,sc} \leq cn^i * volumecapacity - \sum_{j=1}^{t-1} ra^i_{j,sc} \;\; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [10]$$

$$A^i_{t,sc} \leq v_{t,sc} - \sum_{j=i+1}^{B} ra^j_{t,sc} \; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [11]$$

$$cn^i * volumecapacity - \sum_{j=1}^{t-1} ra^i_{j,sc} - A^i_{t,sc} \leq volumecapacity\left(1 - z^i_{t,sc}\right) for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [12]$$

$$v_{t,sc} - A^i_{t,sc} - \sum_{j=i+1}^{B} ra^j_{t,sc} \leq \max(v_{t,sc}) z^i_{t,sc} \;\; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [13]$$

$$ra^i_{t,sc} \leq v_{t,sc}\, x^i_{t,sc} \; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \;\; [14]$$

$$ra^i_{t,sc} - a^i_{t,sc} \leq \max(v_{t,sc})\left(1 - x^i_{t,sc}\right) for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \; [15]$$

$$a^i_{t,sc} - ra^i_{t,sc} \leq \max(v_{t,sc})\left(1 - x^i_{t,sc}\right) for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \; [16]$$

$$\sum_{i=1}^{B} cn^i = 1 \; [17]$$

$$EC_{it} \geq EC_{i-1,t} \; for\; i = 2,\dots,B, for\; t = 1,\dots,T \; [18]$$

$$x_{t,sc} \geq \frac{1}{B}\sum_{i=1}^{B} x^i_{t,sc} \;\; for\; t = 1,\dots,T, sc = 1,\dots,N \; [19]$$

$$x_{t,sc} \leq \sum_{i=1}^{B} x^i_{t,sc} \;\; for\; t = 1,\dots,T, sc = 1,\dots,N \; [20]$$

$$cn^i * volumecapacity - \sum_{j=1}^{t-1} ra^i_{j,sc} - ca^i_{t,sc} \leq volumecapacity * (1 - y^{3i}_{t,sc}) \; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \; [21]$$

$$ca^i_{t,sc} \leq cn^i * volumecapacity - \sum_{j=1}^{t-1} ra^i_{j,sc} \;\; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \; [22]$$

$$ca^i_{t,sc} \leq volumecapacity * y^{3i}_{t,sc} \; for\; i = 1,\dots,B, for\; t = 1,\dots,T, sc = 1,\dots,N \; [23]$$

MP 3: MIP for strategy 2b

Domain constraints:

$EC_{it}, ra^i_{t,sc}, ca^i_{t,sc}, cn^i,\ A^i_{t,sc} : \geq 0\ for\ i = 1, \dots, B, t = 1, \dots, T, sc = 1, \dots, N$ [24]

$y^1_{t,sc}, y^2_{t,sc}, y^{3i}_{t,sc}, x^i_{t,sc},, x_{t,sc}, z^i_{t,sc} \in \{0,1\}\ for\ i = 1, \dots, B, t = 1, \dots, T, sc = 1, \dots, N$ [25]

Constraints 2 and 3 set the first variable ($y^1_{t,sc}$) to 1 if there is still enough weight capacity remaining, else to 0. Constraints 4 and 5 set the second variable ($y^2_{t,sc}$) to 1 if there is still enough volume remaining in the buckets for which the EC is smaller than or equal to the request's SCb/m^3, else to 0. Constraints 6 and 7 set the third variable for the *i*-th bucket ($y^{3i}_{t,sc}$) to 1 if the request meets that bucket's SCb/m^3, else to 0. Constraints 8 and 9 make sure that a request is accepted in a bucket when all three conditions for that bucket are met, and rejected when less than three conditions are met. Constraints 10 to 13 set auxiliary variables $A^i_{t,sc}$ equal to $\min(remaining\ volume,\ v_{t,sc} - \sum_{j=i+1}^{B} ra^j_{t,sc})$. Constraints 14 to 16 set the accepted volume in each bucket equal to $A^i_{t,sc}$ if the request is accepted, and equal to 0 if the request is rejected. Constraint 17 makes sure the bucket proportions add up to 1. Constraints 18 makes sure that the EC for each bucket is not lower than the EC for the preceding buckets. Constraints 19 and 20 set auxiliary variables $x_{t,sc}$ equal to 1 if the request is accepted in any of the buckets, 0 otherwise. Constraints 21 to 23 set auxiliary variable $ca^i_{t,sc}$ equal to the remaining volume in bucket *i* if the EC of that bucket is satisfied, 0 otherwise.

Note that this program can easily be used in the case of s2a as well by dropping the subscript *t* in the variable EC.

In this chapter, mathematical models for determining the optimal decision variables for the various bid-price strategies have been formalized. In later chapters, the performance of the various bid-price strategies will be analyzed in real-life cases using the models described in this chapter.

Besides these strategies, in the next chapter the general ideas used in the dynamic programming (DP) algorithms for the accept/reject decision presented in the literature review section will be applied in a custom algorithm that is most appropriate for the application at hand.

CHAPTER 6 DYNAMIC PROGRAMMING ALGORITHM

In this chapter, the dynamic programming algorithm used in this thesis will be presented. The basic dynamic programming algorithm will assume the following problem structure:

- A single flight with known, deterministic capacities, possibly including overbooking.
- Capacities are in three dimensions: weight, volume and pallet positions.
- Deterministic number of requests, known upfront.
- Deterministic SCb and capacity requirements for all requests, known upfront.
- An offloading cost function, that takes as input the amount of overbooked capacities used, in all dimensions and gives as output the expected total offloading costs for these overbooked capacities used.

Extensions of the basic model include:

- Including multiple flights.
- Including expected show-up rates for requests.

These extensions will be discussed after having formalized the basic algorithm.

Regarding the offloading function, the following clarification needs to be made. Offloading costs can be seen as a combination of the physical and organizational costs of having to store the associated offloaded cargo for a longer amount of time and the costs associated with compensating the customer for the delay in transportation. It is assumed that the customer does not cancel their request for transportation, but that the associated offloaded cargo will be delivered by the next appropriate flight. This is common practice in air cargo. Thus, we do not need to take into account that the SCb that was incurred for accepting the request needs to be subtracted again. The SCb is still earned by the airline. However, we do need to subtract the request's capacity requirements from the total capacities of the next flight. Determining exactly which cargo to offload is an optimization problem on its own and is out of scope of this thesis. Including this optimization problem would make the DP-algorithm very unwieldy. Because the exact offloading costs will depend on the exact type of cargo that is offloaded and the offloading cost function only takes as input the amount of overbooked capacities used in each dimension, the offloading cost function is appropriate and thus the related costs are 'expected costs'.

The following table introduces the notation used in the basic version of the DP-algorithm.

Table 2: Notation used in basic DP-algorithm

Notation	*Explanation*
B_t^j	Accepted capacity regarding dimension *j* in period *t*, if no booking was accepted in period *i*, B_t^j equals 0 for all *j*
E_j^+	The amount of overbooked capacity of dimension *j* that is actually used
O(v, w, p)	Offloading costs when having to offload *v* volume, *w* weight and *p* positions
P	Real amounts of positions on the flight plus any overbooked capacity if applicable
P_R	Real amount of pallet positions on the flight excluding overbooking
p_t	Number of remaining pallet positions in period *t*
$R(t, v_t, w_t, p_t)$	Optimal total remaining SCb from time *t* to 0, where 0 is the time of departure, given v_t remaining volume, w_t remaining weight and p_t remaining pallet positions in the flight at time *t*
s_t	SCb of request in period *t*
T	Period in which the first booking on the flight could be placed. 0 is period of departure
t	Index for period
V	Operational volume capacity of the flight plus any overbooked capacity if applicable
V_R	Operational volume capacity on flight excluding overbooking
v_t	Remaining volume in period *t*
W	Real weight capacity of the flight plus any overbooked capacity if applicable
W_R	Real weight capacity on flight excluding overbooking
w_t	Remaining weight in period *t*

The dynamic programming algorithms presented in the papers (cf. Amaruchkul, et al. (2007), Huang & Hsu (2005) and Papastavrou, et al. (2007)) generally work as follows: first an

optimal expected remaining SCb function R[2] is defined, where:

Total SCb = Revenue from all accepted offers – variable costs associated with flight – offloading costs if applicable; more precisely,

R(t, v_t, w_t, p_t) = optimal total remaining SCb from time t to 0, where 0 is the time of departure, given v_t remaining volume, w_t remaining weight and p_t remaining pallet positions in the flight at time t. Note that $v_t = V - \sum_{i=t+1}^{T} B_i^v$, where V equals the operational volume capacity of the flight plus any overbooked capacity if applicable, T equals the period in which the first booking could be placed and B_i^j is defined as the accepted capacity regarding dimension j in period i. If no booking was accepted in period i, B_i^j equals 0 for all j. Equivalently, $w_t = W - \sum_{i=t+1}^{T} B_i^w$, where W equals the real weight capacity of the flight plus any overbooked capacity if applicable and $p_t = P - \sum_{i=t+1}^{T} B_i^p$, where P equals the real amounts of positions on the flight plus any overbooked capacity if applicable.

Note that the time between successive periods can be made arbitrarily small by making the unit of time more granular and by increasing T. This way, the problem can be modeled in such a way that in any period at most one booking is requested.

The optimal total remaining SCb function has to be a non-decreasing function of t, v_t and w_t, keeping the other two variables fixed, as it is never unbeneficial to have more time, volume or weight available.

Potential offloading costs are incurred at the period of departure and the DP variable is initialized here:

$R(0, v_0, w_0, p_0) = -O(V - V_R - v_0, W - W_R - w_0, P - P_R - p_0)$, for $0 \leq v_0 \leq V, 0 \leq w_0 \leq W, 0 \leq p_0 \leq P$ where O is some expected offloading cost function to be specified based on practical considerations and V_R, W_R and P_R are the operational volume capacity, real weight capacity of the flight and real amount of pallet positions on the flight (excluding overbooking). Note that $V - V_R - v_0$, $W - W_R - w_0$ and $P - P_R - p_0$, when positive, equal the amount of overbooked volume or weight capacity that is used by the flight. If all are negative, expected offloading costs may be assumed to be close to zero.

[2] See Appendix A for an overview of the notation introduced in this chapter.

A general recursive relation for $R(t, v_t, w_{t,}\, p_{t,})$ can be devised as follows:

$$R(t, v_t, w_{t,}\, p_{t,}) = \max\Big(s_t + R(t-1, v_t - B_t^v, w_t - B_t^w,\, p_t - B_t^p), R(t-1, v_t, w_{t,} p_{t,})\Big), for\ 1 \le t \le T, 0 \le v_t \le V, 0 \le w_t \le W, 0 \le p_t \le P$$

Equation 2

where s_t equals the shipment contribution associated with accepting the offer at time t. Whenever $R(t, v_t, w_{t,}\, p_t)$ needs to be evaluated in this recursive relation, and v_t, w_t and/or p_t is negative, $R(t, v_t, w_{t,}\, p_t)$ should be set to $-\infty$. This prevents booking more weight and volume than the planned capacities including overbooking.

In this recursive equation, the first element in the maximum operator is related to the decision of accepting the offer at period *t*, and the second element is related to the decision of rejecting the offer. Thus, the decision rule whether to accept an offer in period *t* is as follows: accept if $s_t \ge R(t-1, v_t, w_{t,}\, p_t) - R(t-1, v_t - B_t^v, w_t - B_t^w,\, p_t - B_t^p)$.

Thus, using this approach, one can both calculate whether to accept a certain booking request and calculate optimal time-dependent bid-prices, which are equal to $R(t-1, v_t, w_t,\, p_t) - R(t-1, v_t - B_t^v, w_t - B_t^w,\, p_t - B_t^p)$.

It should be noted that in practice the amount of pallet positions on planes is rather small (in the range 10-25). Thus, the inclusion of the amount of pallet positions as a capacity dimension does not dramatically increase the amount of states of the DP-variable R(t, v_t, w_t, p_t).

The DP variable, i.e. the optimal revenue can also be determined in a much simpler manner, using a BIP formulation:

$R(T, v_t, w_t, p_t)$ = Max $\sum_{t=1}^{T} s_t x_t - O(E_v^+, E_w^+, E_p^+)$ s.t.

$\sum_{i=1}^{T} x_i q_i^j \leq w_t \; \forall \; j \in \{w, v, p\}$

$E_v = \mathrm{V} - \mathrm{V_R} - v_t + \sum_{i=1}^{T} x_i q_i^v$

$E_w = \mathrm{W} - \mathrm{W_R} - w_t + \sum_{i=1}^{T} x_i q_i^w$

$E_p = \mathrm{P} - \mathrm{P_R} - p_t + \sum_{i=1}^{T} x_i q_i^p$

$E_j = E_j^+ - E_j^- \; \forall \; j \in \{w, v, p\}$

$E_j^+ \geq 0 \; \forall \; j \in \{w, v, p\}$

$E_j^- \geq 0 \; \forall \; j \in \{w, v, p\}$

$x_i \in \{0,1\}, for \; 1 \leq i \leq T$

MP 4: MIP for knapsack problem with offloading function

where x_i is a binary variable equaling 1 if the booking request in period *i* is accepted, q_i^j is dimension *j*'s capacity requirement of period *i*'s booking request. Note that E_j^+ equals the amount of overbooked capacity of dimension *j* that is actually used. As long as the offloading function O is linear in the positive components of the overbooked capacities, i.e., linear in E_{vf}^+, E_{wf}^+ and E_{pf}^+, this is a linear integer program which can be approximated well by conventional Branch and Bound-algorithms. In the case of an offloading function that is non-linear in the positive components of the overbooked capacities, this program might become difficult to solve. In the case of a linear offloading function, the BIP-formulation should be used. In this case, when using a commercial solver, the running times when using this BIP-formulation are much faster than using the general DP-formulation.
Although the above DP algorithm is promising, the issue of deterministic demand still needs to be addressed. Obviously, in a real-life case, future demand is not exactly known. However, we can assume that the distributions of weight, volume, pallet positions and SCb of requests are known.

The approach that will be used in the research of this thesis, is to generate a large amount of lists of future requests and determine in each time period whether to accept or reject the request at that time by solving the DP-algorithm for all demand instances and accepting when the average cumulative SCb when accepting the current request is larger than the expected cumulative SCb when rejecting the current request. This approach will be

formalized at the end of this chapter. Note that the DP-algorithm itself does not change. It is just applied a large number of times with each time a different set of generated future demand. First, two extensions of the basic DP-algorithm will be discussed. After having discussed the extensions, the final algorithm including these extensions will be given.

Extension 1: allowing booking requests involving multiple flights

Up to this moment, booking requests only involved one flight. Now, the algorithm will be extended to also be able to incorporate booking requests involving multiple flights, with each their own capacities. Conceptually, this factor is not difficult to remodel in the discussed DP algorithm. The volume and weight capacities of all the additional flights can simply be considered as additional capacity dimensions. Thus, we can reformulate the DP variable as follows:

R($\bigcup_{f\in F}[t_f, v_{tf}, w_{tf}, p_{tf}]$), where *F* is the set of flights. Also, we need to augment B_t^j to B_{tf}^j, for all capacity dimensions *j*. The change of the recursion equation is obvious. This way, given enough computation time and resources, the problem can still be solved to optimality. Obviously, a major problem is the extremely high dimensionality of this approach, in the case of a larger amount of flights. Another practical consideration is which flights to include. The booking request at a certain period *t* concerns a certain set of flights, however, future booking requests might include some of the flights in this set as well as other flights, which means these additional flights need to be included in the state space as well. However, if we have a number of disjoint sets of flights we can simply decompose the problem in smaller problems. For instance, if all booking requests involve flights from either set A or set B, where A ∩ B =∅, R($\bigcup_{f\in A\cup B}[t_f, v_{tf}, w_{tf}, p_{tf}]$) = R($\bigcup_{f\in A}[t_f, v_{tf}, w_{tf,} p_{tf}]$) + R($\bigcup_{f\in B}[t_f, v_{tf}, w_{tf,} p_{tf}]$) for all states.

Extension 2: incorporating expected show-up rates for requests:

In the original DP algorithm, show-up rates are incorporated by overbooking a certain percentage of capacity. Then, in the initialization of the DP variable, expected costs of offloading are calculated based on the actually used amount of overbooking. However, this approach is rooted in the assumption that show-up rates are approximately uniform over all bookings. In practice, this is often not the case. Often, requests including multiple routes involve low-demand routes that are included in the deal to make up for higher capacity commitment on high-demand routes. However, often flight analysts suspect that the customer will have a bad show-up rate in the low-demand flight. Thus, the offloading cost

function should be adjusted so that it takes into account the expected show-up rates of the various booking requests accepted in the past. Note that the show-up rate is not a probability. The outcome is not binary in the sense that the customer shows up with his requested capacities or not, but the customer shows up with a certain percentage of the requested capacities.

Final dynamic programming algorithm:

In this section, the final DP-model will be developed, incorporating the two extensions discussed.

In order to include the show-up rates of requests, the DP-variable is augmented with states C_{tf}^{v}, C_{tf}^{w} and C_{tf}^{p} where C_{tf}^{j} equals the cumulative expected capacity of dimension *j* that will show up of all bookings booked from time *T* to *t+1*. Thus, $C_{tf}^{j} = \sum_{i=t+1}^{T} Pr_{if} B_{if}^{v}$, where Pr_{if} equals the showup-rate of the request in period *i*, assumed constant over all dimensions *j*.

The following basic DP-model will be used for analyzing the problem at hand. All booking requests for all periods are assumed known in advance, including their capacity requirements and show-up rates, either because we want to analyze certain accept/reject-strategies a posteriori or by drawing the requests from appropriate distributions.

General DP-formulation to be used, including show-up rates

$$\begin{cases} R_f(0, v_{0f}, w_{0f}, p_{0f}, C_{0f}^{v}, C_{0f}^{w}, C_{0f}^{p}) = -\mathrm{O}(C_{0f}^{v} - \mathrm{V_{Rf}}, C_{0f}^{w} - \mathrm{W_{Rf}}, C_{0f}^{p} - \mathrm{P_{Rf}}), \text{for } 0 \le v_{0f} \le V_f, 0 \le w_{0f} \le W_f, 0 \le p_{0f} \le P_f \\ R_f(t, v_{tf}, w_{tf}, p_{tf}, C_{tf}^{v}, C_{tf}^{w}, C_{tf}^{p}) = \max \begin{pmatrix} s_{tf} + R(t-1, v_{tf} - B_{tf}^{v}, w_{tf} - B_{tf}^{w}, p_{tf} - B_{tf}^{p}, C_{tf}^{v} + Pr_{tf} B_{t}^{v}, C_{tf}^{w} + Pr_{tf} B_{t}^{w}, C_{tf}^{p} + Pr_{tf} B_{t}^{p}), \\ R(t-1, v_{tf}, w_{tf,}, p_{tf}, C_{tf}^{v}, C_{tf}^{w}, C_{tf}^{p}) \end{pmatrix}, \\ for\ 1 \le t \le T, 0 \le v_{tf} \le V_f, 0 \le w_{tf} \le W_f, 0 \le p_{tf} \le P_f. \\ R_f(t, v_{tf}, w_{tf}, p_{tf}\ C_{tf}^{v}, C_{tf}^{w}, C_{tf}^{p}) = -\infty\ if\ v_{tf} < 0\ or\ w_{tf} < 0\ or\ p_{tf} < 0 \end{cases}$$

Equation 3

Note that we start evaluating the R-function for the states that have as input time period the period of departure. At this time, there will be no more booking requests, so the remaining SCb will never be positive. At the time of departure, only the expected offloading costs still need to be considered. This leads to the first equation. The three input variables for the offloading cost function are the amount of overbooked volume, weight and positions, respectively.

For any other input time period, the second equation is applicable. In this case, a decision

needs to be made whether the current booking request will be accepted or rejected. The first argument of the maximum function refers to the case of accepting the current request, in which case we reduce the available capacities, increase the cumulative expected show-up quantities and have optimal remaining SCb consisting of the sum of the SCb of the current request and the optimal remaining SCb with the reduced remaining capacities and the increased booked quantities when the next request arrives. The second argument refers to the case that we reject the request, in which case optimal remaining SCb equals the optimal remaining SCb at the time of the next request with the current remaining capacities and cumulative expected show-up quantities. Obviously, the optimal decision is associated with the maximum of these two arguments.

The third equation ensures that overbooking limits are respected.

When the input time period equals *T*, all feasible combinations of accepting/rejecting the future booking requests have been considered and $R_f(T, v_{tf}, w_{tf}, p_{tf}, C_{tf}^v, C_{tf}^w, C_{tf}^p)$ equals the optimal remaining SCb we are interested in. This optimal remaining SCb is associated with the optimal combination of accepting/rejecting the future booking requests.

The decision rule is to accept the offer in period *t* involving a single flight *f* if $s_{tf} \geq R(t-1, v_{tf}, w_{tf,} p_{tf,} C_{tf}^v, C_{tf}^w, C_{tf}^p) - R\left(t-1, v_{tf} - B_{tf}^v, w_{tf} - B_{tf}^w, p_{tf} - B_{tf}^p, C_{tf}^v + Pr_{tf}B_{tf}^v, C_{tf}^w + Pr_{tf}B_{tf}^w, C_{tf}^p + Pr_{tf}B_{tf}^p\right)$.

Thus, using this formulation we can retrieve the optimal booking decisions for all periods recursively.

Note that we can evaluate $R_f(t, v_{tf}, w_{tf}, p_{tf,} C_{tf}^v, C_{tf}^w, C_{tf}^p)$ for any positive integer *t* by first evaluating $R_f(0, v_{0f}, w_{0f}, p_{0f}, C_{0f}^v, C_{0f}^w, C_{0f}^p)$ for all values $0 \leq v_{0f} \leq V_f, 0 \leq w_{0f} \leq W_f, 0 \leq p_{0f} \leq P_f, 0 \leq C_{0f}^v \leq 1,\ 0 \leq C_{0f}^w \leq 1, 0 \leq C_{0f}^p \leq 1$ with appropriate increments in the variables. Then, the period of time can be incremented by one and evaluated for all values $0 \leq v_{0f} \leq V_f, 0 \leq w_{0f} \leq W_f, 0 \leq p_{0f} \leq P_f, 0 \leq C_{0f}^v \leq 1,\ 0 \leq C_{0f}^w \leq 1, 0 \leq C_{0f}^p \leq 1, \text{again, after which}$ the period of time can be incremented by one again. When the period of time equals *t*, $R_f(t, v_{tf}, w_{tf}, p_{tf,} C_{tf}^v, C_{tf}^w, C_{tf}^p)$ can be evaluated.

However, this is not an efficient approach as the expected remaining SCb function does not need to be evaluated for all these states in order to evaluate $R_f(t, v_{tf}, w_{tf}, p_{tf,} C_{tf}^v, C_{tf}^w, C_{tf}^p)$. For example, take a hypothetical booking period consisting of two periods and a flight having 10 volume. The booking request in the first period, where *t* = 2, comprises 6 volume and the booking request in the second period, where *t* = 1, comprises 3 volume. In this case, we do

not need to evaluate the R-function for the complete interval of $v_{tf} \in [0, 10]$, for $t \in \{0, 1, 2\}$, as evaluating for v_{2f} = 10, v_{1f}= 10, v_{1f} = 4, v_{0f} = 10, v_{0f} = 4 and v_{0f} = 1 suffices. Generalizing this idea, see Appendix C for pseudocode of algorithm S_f^1 that finds all relevant states for which it is needed to evaluate the R-function.

A serious problem is the high number of states for larger values of *t*. As from any state the remaining SCb of two other states is needed, an upper bound on the number of states is $\sum_{i=0}^{t} 2^i$. The easiest way to reduce the number of states is to discretize the state space and rounding the original valuse to an appropriate amount of decimals. Thus, we decrease the granularity of the state space by discretizing it. For this, we choose v_{tf}^r = round($10^i \frac{v_{tf}}{V_f}$)/10^i, w_{tf}^r = round($10^j \frac{w_{tf}}{W_f}$)/10^j, p_{tf}^r = round($10^k \frac{p_{tf}}{p_f}$)/10^k, $C_{tf}^{vr} = \text{round}(10^l \frac{C_{tf}^v}{V_f})/10^l$, $C_{tf}^{wr} = \text{round}(10^m \frac{C_{tf}^w}{w_f})/10^m$ and $C_{tf}^{pr} = \text{round}(10^n \frac{C_{tf}^p}{p_f})/10^n$ where *i*, *j*, *k*, *l*, *m* and *n* are the number of decimals to be rounded to. Let S = *i+j+k+l+m+n*, the sum of the rounding constants. This gives an upper bound on the number of states of (t+1)$(10+1)^S$. Due to the rounding, $v_{tf} \neq v_{tf}^r V_f$ but we can guarantee $\left|v_{tf} - v_{tf}^r V_f\right| \leq \frac{V_f}{2*10^i}$, as the maximal absolute error when translating $\frac{v_{tf}}{V_f}$ to $v_{tf}^r$is $\frac{1}{2*10^i}$ For the other rounded variables, an equivalent relationship holds.

Implementing this change, the expected remaining SCb function changes to:

$$R_f\left(0, v^r_{0f}, w^r_{0f,}\, p^r_{0f}, C^{vr}_{0f}, C^{wr}_{0f}, C^{pr}_{0f}\right) = -\mathrm{O}\left(C^{vr}_{0f}\mathrm{V_f} - \mathrm{V_{Rf}}, C^{wr}_{0f}\mathrm{W_f} - \mathrm{W_{Rf}}, C^{pr}_{0f}\mathrm{P_f} - \mathrm{P_{Rf}}\right)$$
$$\text{for } 0 \le C^{vr}_{0f} \le 1, 0 \le C^{wr}_{0f} \le 1, 0 \le C^{pr}_{0f} \le 1$$

$$R_f\left(t, v^r_{tf}, w^r_{tf}, p^r_{tf}, C^{vr}_{tf}, C^{wr}_{tf}, C^{pr}_{tf}\right) =$$
$$\mathrm{I}\left(v^r_{tf} - \frac{B^v_{tf}}{V_f} < 0 \vee w^r_{tf} - \frac{B^w_{tf}}{W_f} < 0 \vee p^r_{tf} - \frac{B^p_{tf}}{P_f} < 0\right) * R\left(t-1, v_{tf}, w_{tf,} p_{tf,} C^v_{tf}, C^w_{tf}\right)$$
$$+ \left(\mathrm{I}\left(v^r_{tf} - \frac{B^v_{tf}}{V_f} \ge 0\right) * \mathrm{I}\left(w^r_{tf} - \frac{B^w_{tf}}{W_f} \ge 0\right) * \mathrm{I}\left(p^r_{tf} - \frac{B^p_{tf}}{P_f} \ge 0\right)\right.$$
$$\left. * \max\left(\begin{array}{c} s_{tf} + R_f\left(\begin{array}{c} t-1, r\left(i, v^r_{tf} - \frac{B^v_{tf}}{V_f}\right), \\ r\left(j, w^r_{tf} - \frac{B^w_{tf}}{W_f}\right), r\left(k, p^r_{tf} - \frac{B^p_{tf}}{P_f}\right) r(l, C^{vr}_{tf} + \frac{Pr_{tf}B^v_t}{V_f}), \\ r(m, C^{wr}_{tf} + \frac{Pr_{tf}B^w_t}{W_f}), r(n, C^{pr}_{tf} + \frac{Pr_{tf}B^p_t}{P_f}) \\ , R_f\left(t-1, v^r_{tf}, w^r_{tf}, p^r_{tf}, C^{vr}_{tf}, C^{wr}_{tf}, C^{pr}_{tf}\right) \end{array}\right) \end{array}\right)\right)$$
$$for\ 1 \le t \le T, 0 \le v^r_{tf} \le 1, 0 \le w^r_{tf} \le 1, 0 \le w^p_{tf} \le 1, 0 \le C^{vr}_{0f} \le 1, 0 \le C^{wr}_{0f} \le 1, 0 \le C^{pr}_{0f} \le 1.$$
$$R_f\left(t, v^r_{tf}, w^r_{tf}, p^r_{tf}, C^{vr}_{tf}, C^{wr}_{tf}, C^{pr}_{tf}\right) = -\infty\ if\ \left(v^r_{tf} < 0\ or\ w^r_{tf} < 0\ or\ p^r_{tf} < 0\right)$$

Equation 4

where I(s) is the binary indicator function returning 1 if statement *s* is true, 0 otherwise, and r(i,x) = $\mathrm{round}(10^i x)\ /10^i$. Algorithm S^2_f in appendix C is the updated version of algorithm S^1_f, implementing this change of discretization of the state space.

Algorithm R^2_f in Appendix C generates expected remaining SCb for all relevant states generated by algorithm S^2_f, amongst which the current state, which is the top row.

Algorithm A^2_f that can be found in Appendix C in pseudocode generates the optimal binary decision vector for all relevant states, i.e. whether we should accept the booking request in this state.

Finally, algorithm D^2_f in Appendix C outputs a T-length binary vector with the *t*-th position equaling 1 if the offer at time *t* should be accepted, 0 otherwise. Thus, as output we have for the list of requests in the case, for each request, whether in the optimal solution we should accept this request or reject it.

Note that if we are only interested in the outcome of the R-function, we only need to apply algorithm S^2_f to generate the state space and use the output matrix as in put for algorithm R^2_f to calculate the outcome of the R-function.

Running times:

In the following table average running times for running S^2_f, R^2_f, A^2_f and D^2_f consecutively for

a certain flight *f* are enumerated for various rounding precisions and various values of *t*. The following testing configuration is used: $v_{0f}^{r} = w_{0f}^{r} = p_{0f}^{r} = 1, C_{0f}^{vr} = C_{0f}^{wr} = C_{0f}^{pr} = 0, \mathrm{V_f} = 100, \mathrm{W_f} = 200, \mathrm{P_f} = 18, \mathrm{V_{Rf}} = 80, \mathrm{W_{Rf}} = 160, \mathrm{P_{Rf}} = 15$. Show-up rates are generated using a uniform distribution between 0.3 and 1 (with increments of 0.01), volumes are integer uniformly distributed between 1 and 25, weights are integer uniformly distributed between 2 and 50, positions are continuously uniformly distributed between 1 and 3 and SCb's are integer uniformly distributed between 1 and 100. These distributions are not realistic but will suffice for testing the effect of increasing *T o*n running times. The rounding precisions are chosen in such a way that they are related to the magnitude of the variables, i.e. lower precision for the position dimension and higher for the volume dimension, where the case of 14 is a very high precision setting and 11 is a lower precision setting, but still sufficient for most purposes. To be precise, in the case of 11: $k = n = 1.2$ and $i = j = l = m = 2.15$. In the case of $S = 14$ these values are multiplied by $\frac{14}{11}$. The times in brackets in the following table are standard errors of the mean.

Table 3: Running times of DP algorithm

T	**Average running time:** **Over 25 runs**	***S***
10	2.81 seconds (0.09 seconds)	11
10	3 seconds (0.08 seconds)	14
13	43.42 seconds (3.06 seconds)	11
13	57.53 seconds (3.38 seconds)	14
15	10.62 minutes (3.19 minutes)	11
15	19 minutes (6.56 minutes)	14

As can be seen, the running time is very sensitive to the magnitude of *T*. Large running times can be reduced somewhat for the same value of *T* by reducing the rounding precision. It should be noted that the rounding precision is not that important in this specific example because most of the variables are already integer valued, resulting in many duplicate states, which are only added and evaluated once anyway. If we set all distributions to be continuously uniformly distributed, the effect of the rounding precision becomes much more profound. Thus, it is confirmed that discretizing the state space is an appropriate idea in reducing the complexity of the algorithms.

Effect of rounding on algorithm precision:

A setting of $T = 13$ shall be used in conjunction with 10 runs to analyze the effect of the discretizing precision on the calculated value s of the R-function. Ten sets of demand instances have been generated and fixed in each run. The parameterization is the same as above, except that all parameters have been multiplied by 1000, except for the show-up rate. The various settings of rounding precision will then be applied on the same demand instance in order to guarantee comparability of the results. First, the revenue will be calculated without discretizing the state space. This means that the value of the R-function in this case equals the optimal revenue with probability 1, as all possible combinations of accepting and rejecting the various requests are evaluated in an non-discretized manner. This means that we can compare the relative difference of the absolute error between the

average optimal revenue over all 10 cases and the average calculated optimal revenue for any given *S* over all 10 cases.

Table 4: Effect of rounding on DP-algorithm

S	*Average calculated optimal revenue*	*Deviation (%) compared to optimum*
No rounding	470492.6	-
26	470492.6	0%
20	470492.6	0%
17	470492.6	0%
14	470492.6	0%
11	469527.8	-0.2%
10	469722.9	-0.2%
9	469920.3	-0.2%
8	469225.1	-0.2%
7	470485.1	-0.002%
6	476421.6	+1.3%
5	467507.7	-0.6%
4	446900.7	-5%
3	471969.4	+0.3%
2	495522.6	+5.3%
1	502680.1	+6.8%
0	502680.1	+6.8%

As can be seen from the above table, for this case, the discretizing of the state space does not have dramatic influences on the calculated expected SCb, even in the case of rounding to integers (*S*=0). However, it should be noted that it is a bigger problem that in some of the

more rigorous discretizations, the calculated optimal expected SCb is greater than the calculated optimal expected SCb in the case of no rounding. This obviously points to an infeasible solution. To give an extreme example of how this is possible: suppose we round remaining capacities to integers. Suppose we start with 10 capacity units and the first request takes 3 units. If this request is accepted, relative capacity reduces to 0.7 but in the rounding case, remaining relative capacity is still 1. Thus, the algorithm can accept the first request without any capacity cost at all, which can obviously lead to infeasible solutions.

Use of dynamic programming model with stochastic demand:

At this moment, a dynamic programming approach has been established that is able to determine optimal cumulative SCb for a given demand instance. Seeing that in real-life, we only know the current request and not the future requests, the following approach will be used to incorporate unknown future demands.
Here, a certain demand instance or scenario of *T* requests is given and we need to decide for each request whether to accept or reject it. However, we only know the requests up to that moment, not the requests that arrive later in time. Also, when we have made an accept/reject decision, that decision is fixed and we may not change that decision at a later moment of time.

1) Generate *N* demand scenarios, each containing *T*-1 requests.
2) Initialize *t* to *T*, *w* to the weight of the flight and *v* to the volume of the flight.
3) For all scenarios, determine optimal SCb of the following two alternatives, considering current remaining capacities and only requests later than time *t*:
   1: Rejecting the current request at *t*
   2: Accepting the current request considering that this will cost some weight and volume capacity
4) If the sum of optimal SCb's over all scenarios of the first alternative is larger than or equal to the sum of optimal SCb's over all scenarios of the second alternative plus , the request at time *t* should be accepted.
5) Reduce *t* by 1, and, if request was accepted, reduce *w* and *v* accordingly.
6) Repeat steps 3 to 5 until a decision has been made for all requests.

Note that the cumulative SCb of this approach does not need to be equal to the optimal cumulative SCb of the instance. The approach is expected to work better if more scenarios are used.

In step 3, in general the DP-algorithm can be used. If the offloading function is linear, in this step the optimal SCb's can also be determined by solving the appropriate knapsack problem, which is much more time-efficient.

In this chapter, a dynamic programming algorithm that can be used for the booking problem was discussed and formalized. This DP-algorithm is an alternative to the various bid-price strategies for the booking problem described in Chapter 4. A basic version of the algorithm was first described and after this, extensions were discussed which also can handle multiple flights, individual show-up rates for each requests and discretization of the state space in order to speed up calculation, in lieu of some precision. Also, it was discussed how the algorithm, which is deterministic in nature, can be used in stochastic problems where we do not exactly know which requests will arrive in the future. In the next chapter, the data to be used in an investigation into the merits of various strategies for the booking problem will be described. In the chapters after the next, the performance of the various bid-price strategies vis-à-vis each other and vis-à-vis the DP-algorithm will be analyzed.

Chapter 7 Data Collection and Analysis

In the previous chapters, various strategies for the booking problem have been formalized. These strategies include a static bid-price strategy, a time-dependent bid-price strategy, a bucket strategy with static bid-prices and a bucket strategy with time-dependent bid-prices. The performance of the various in terms of maximizing expected cumulative SCb will be investigated using real-life data from AF-KL Cargo. In this chapter the datasets will be described.

Real-life data was collected on two routes, which will be called R1 and R2. These routes are interesting to analyze because flights on these routes are often constrained, making the EC-setting more interesting, i.e. on non-constrained flights it is optimal to set EC equal to 0. All carried cargo on these two routes in June and July 2013 has been stored, including the weight, volume, SCb and date of the request. This last piece of information allows analysis of the time element in bookings, e.g. whether early bookings differ significantly from late bookings. The information related to one piece of cargo including its weight, volume, SCb and date is stored in a document called an air-way bill (AWB). Some descriptive statistics are summarized in the following table:

Table 5: Statistics of two routes

	R1	R2
Number of AWB's	3905	2825
Average weight	904	1518
Std. dev. weight	1628	2861
Coefficient of variation weight	1.80	1.88
Minimum weight	20.42	16.82
(1/6)*100 percentile weight	23.5	29.2
(2/6)*100 percentile weight	35.3	148.6
Median weight	460.6	453.1
(4/6)*100 percentile weight	844.1	1027.3
(5/6)* 100 percentile weight	1548.3	2104.6

Maximum weight	31724	38344
Average volume	4.00	9.32
Std. dev. volume	7.76	17.3
Coefficient of variation volume	1.94	1.86
Minimum volume	0	0.184
(1/6)*100 percentile volume	0.155	0.186
(2/6)*100 percentile volume	0.210	1.235
Median volume	2.32	3.141
(4/6)*100 percentile volume	3.60	5.62
(5/6)*100 percentile volume	5.9	17.24
Maximum volume	186.7	230
Average SCb/m3	387	333
Std. dev. SCb/m3	214	160
Coefficient of variation SCb/m3	0.55	0.48
Minimum SCb/m3	28	-6
(1/6th)*100 percentile SCb/m3	183	183
(2/6th)*100 percentile SCb/m3	213	236
Median SCb/m3	295	298
(4/6th)*100 percentile SCb/m3	576	400
(5/6th)*100 percentile SCb/m3	648	488
Maximum SCb/m3	1819	2118

These distributions can be visualized in a number of boxplots and histograms.

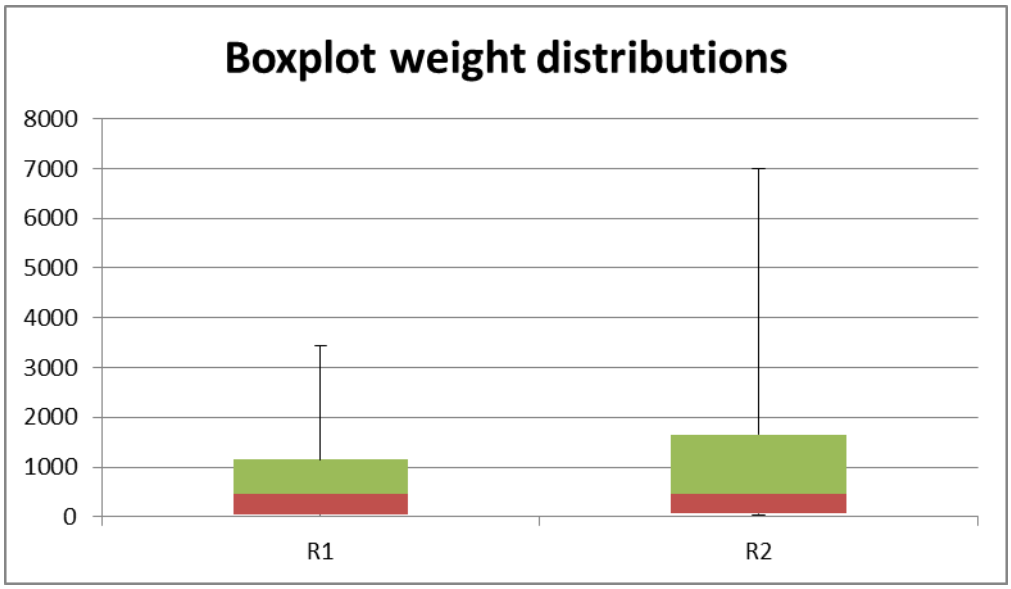


Figure 2: Boxplot weight distributions

Note: top whisker corresponds to 95th percentile instead of maximum for clarity reasons.

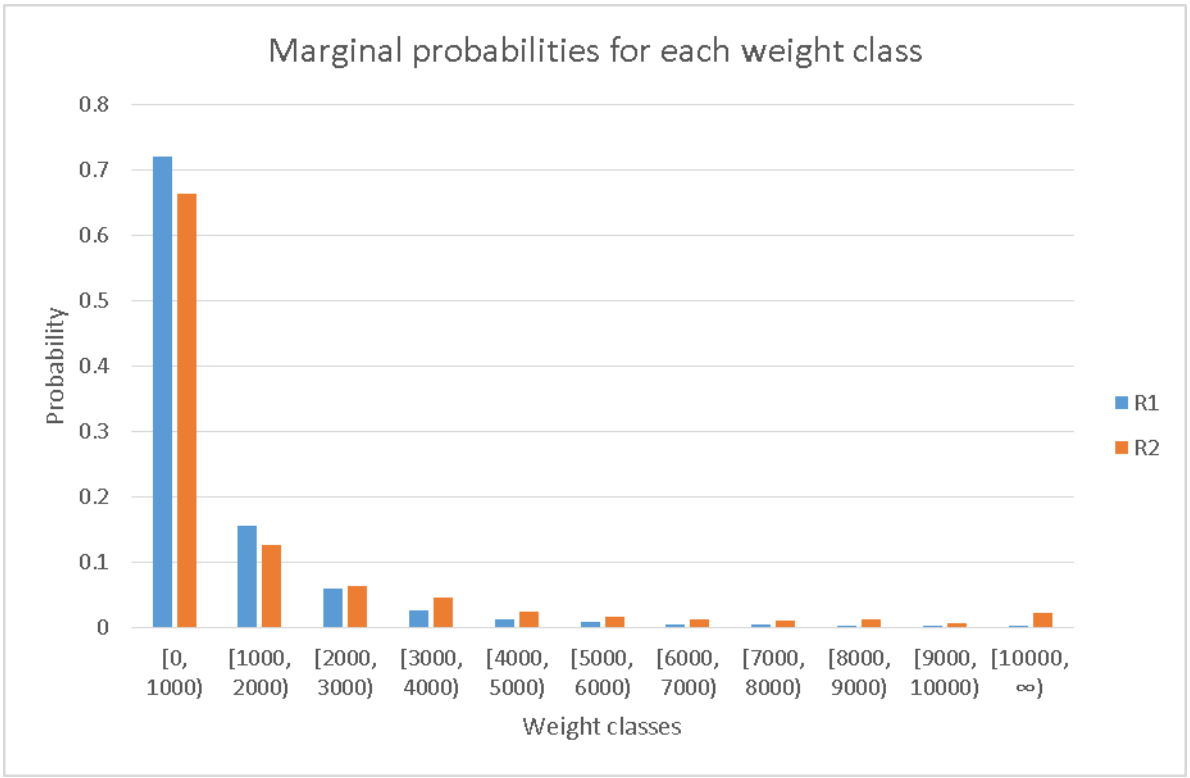


Figure 3: Marginal probabilities for each weight class

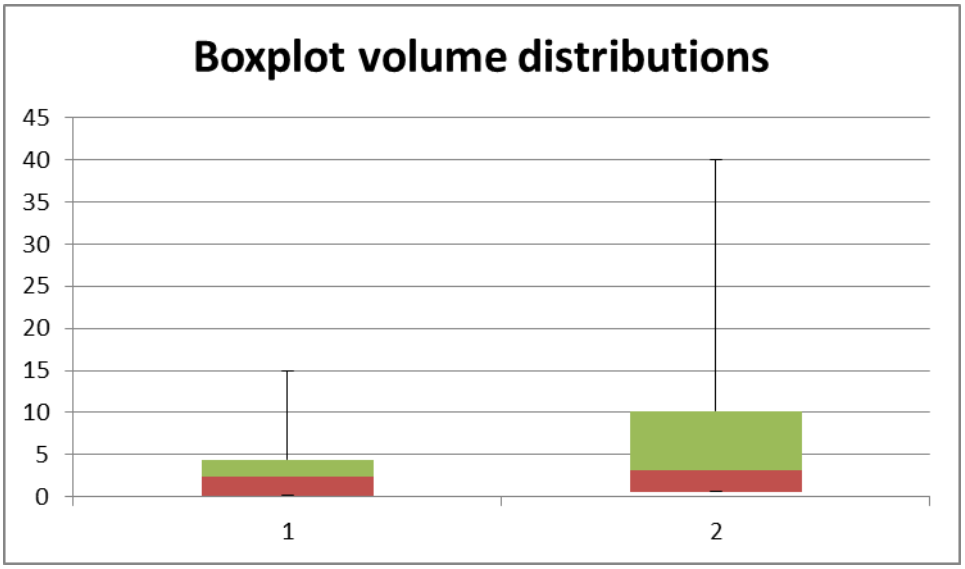


Figure 4: Boxplot volume distributions

Note: top whisker corresponds to 95th percentile instead of maximum for clarity reasons.

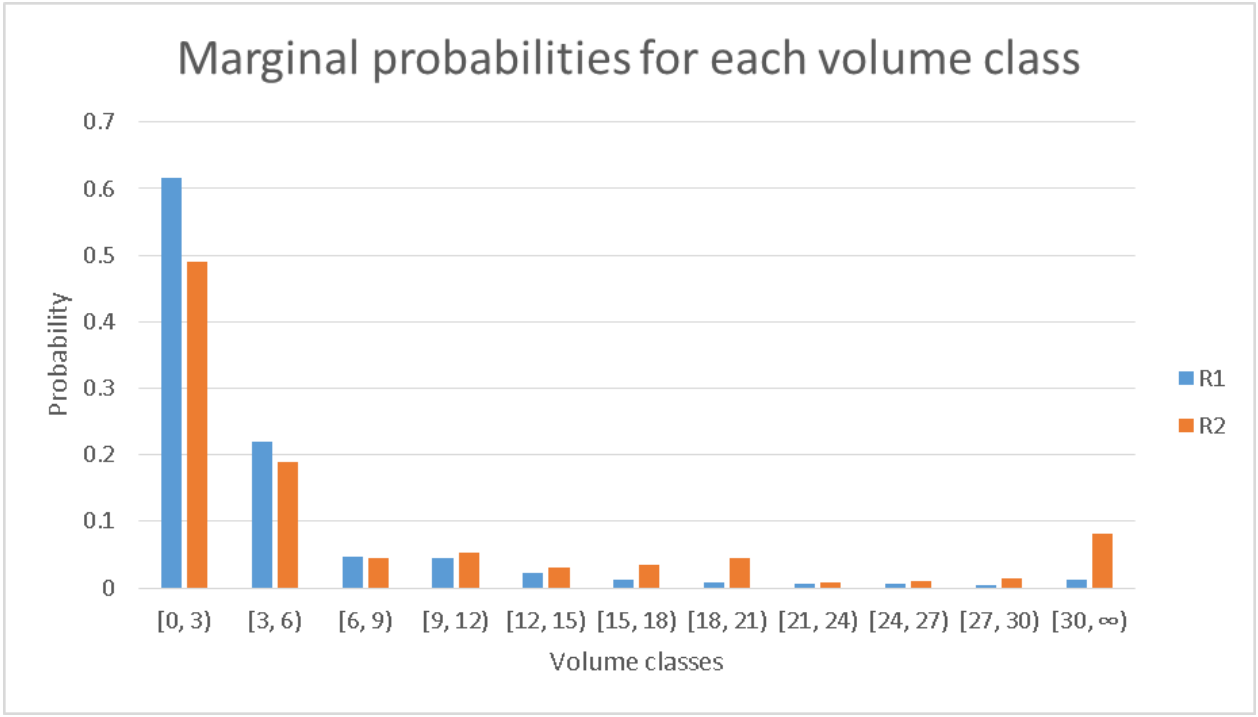


Figure 5: Marginal probabilities for each volume class

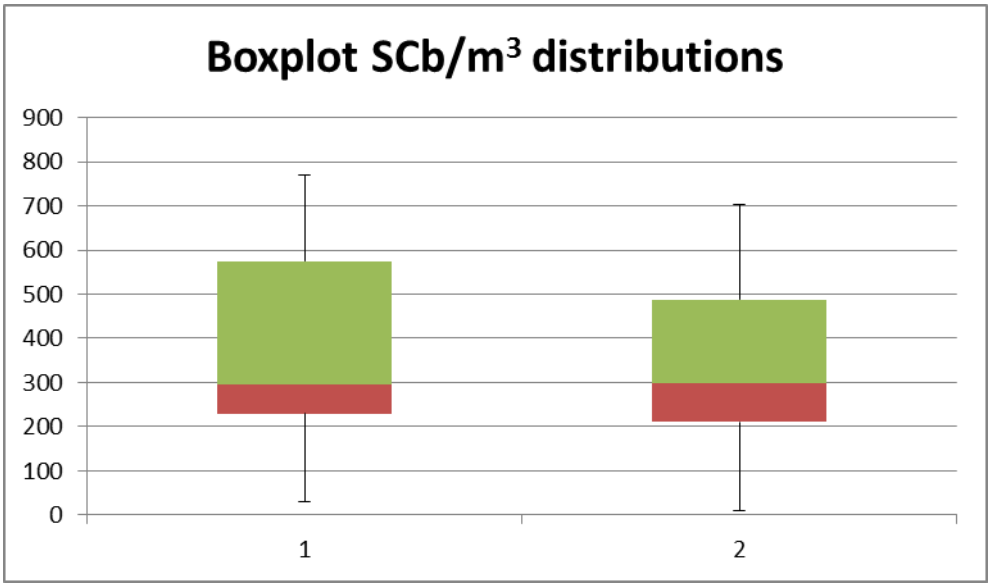


Figure 6: Boxplot SCb/m³ distributions

Note: top whisker corresponds to 99th percentile instead of maximum for clarity reasons.

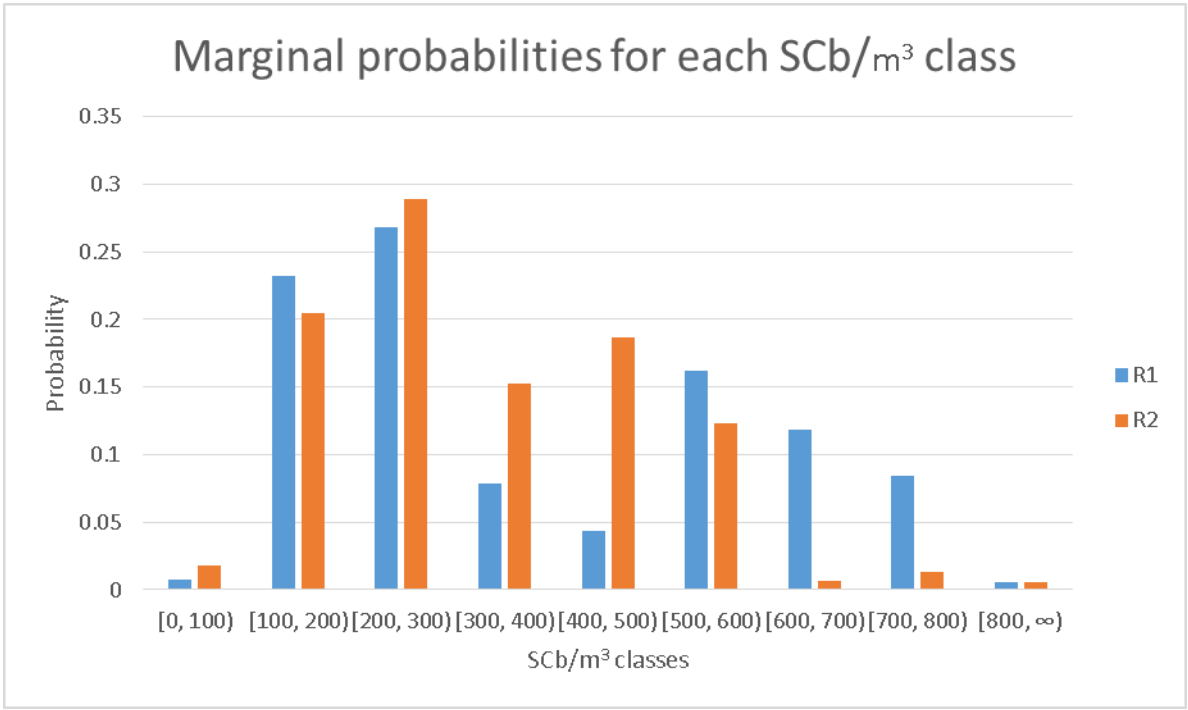


Figure 7: Marginal probabilities for each SCb/m³ class

As can be seen from above statistics and plots, the variation of the size (volume/weight) of requests is quite large. However, the interquartile distance is not that big compared to the total spread. There is a very large group of small shipments and a small group of relatively extremely big shipments. The distribution of SCb/m³ is much less volatile.

As discussed before, the date of a request is also taken into account. The booking window for standard requests opens thirteen days before departure. It is also interesting to analyze the differences in requests' distributions per booking day in these thirteen days before departure. Then we can see if there is any difference in distributions, which would make more sophisticated manners of accepting/rejecting requests more likely to perform better.

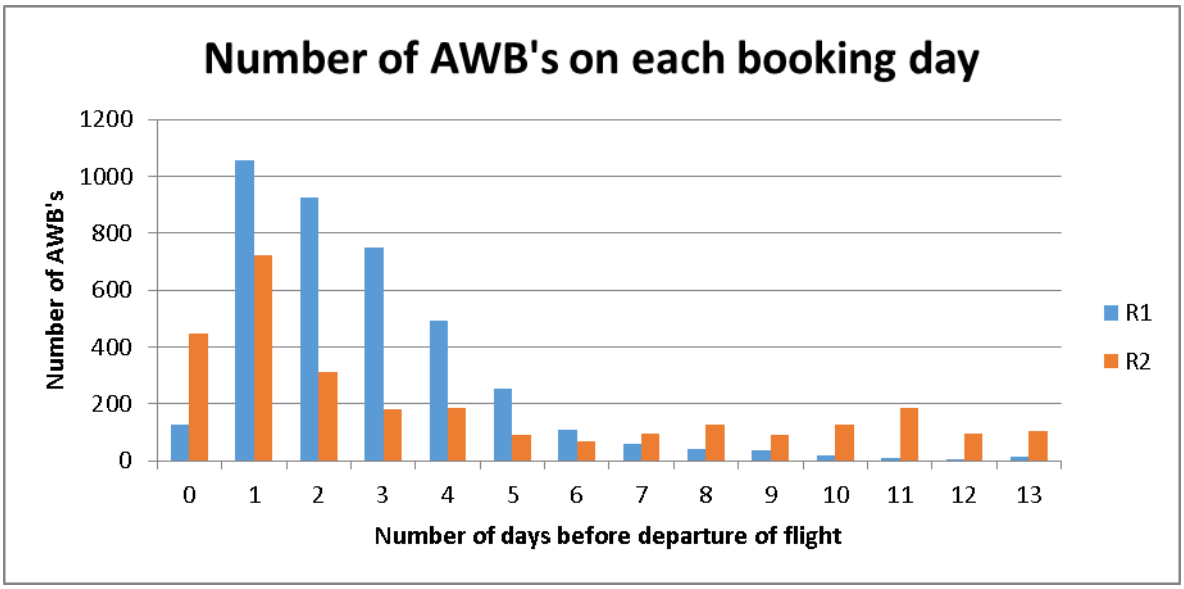


Figure 8: Number of AWB's for each route and booking day

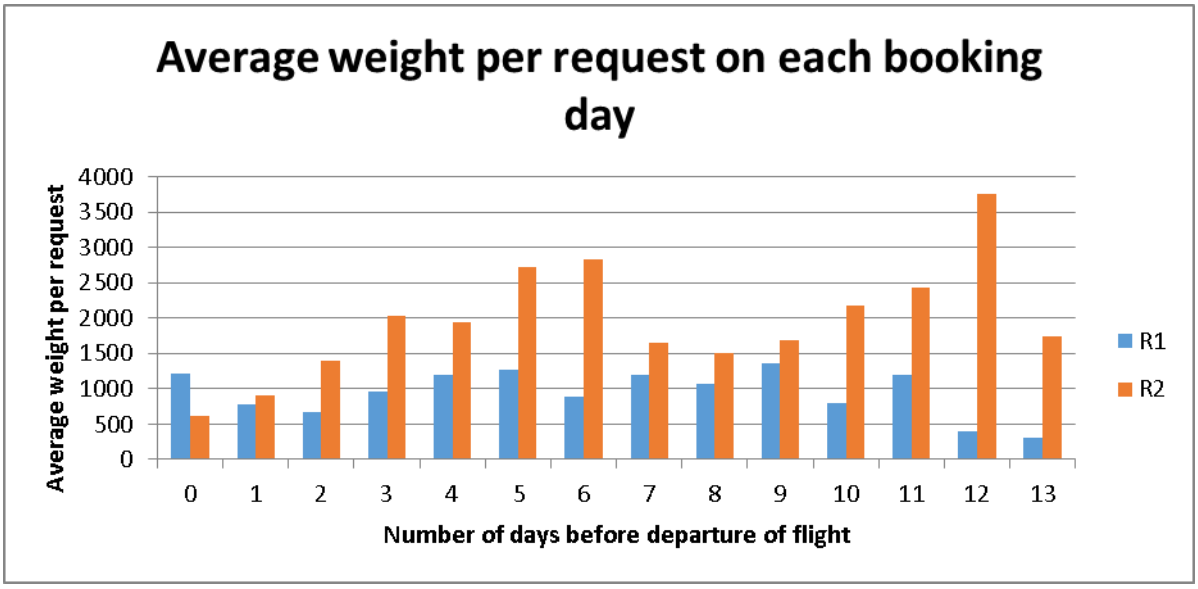


Figure 9: Average weight per request, for each route and booking day

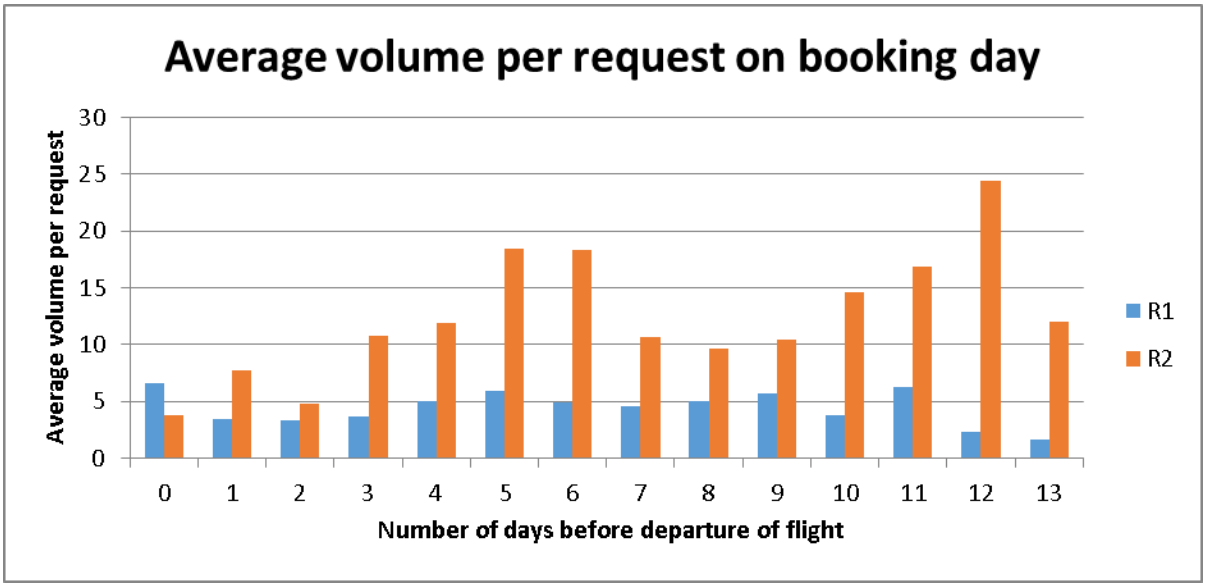


Figure 10: Average volume per request, for each route and amount of days before departure

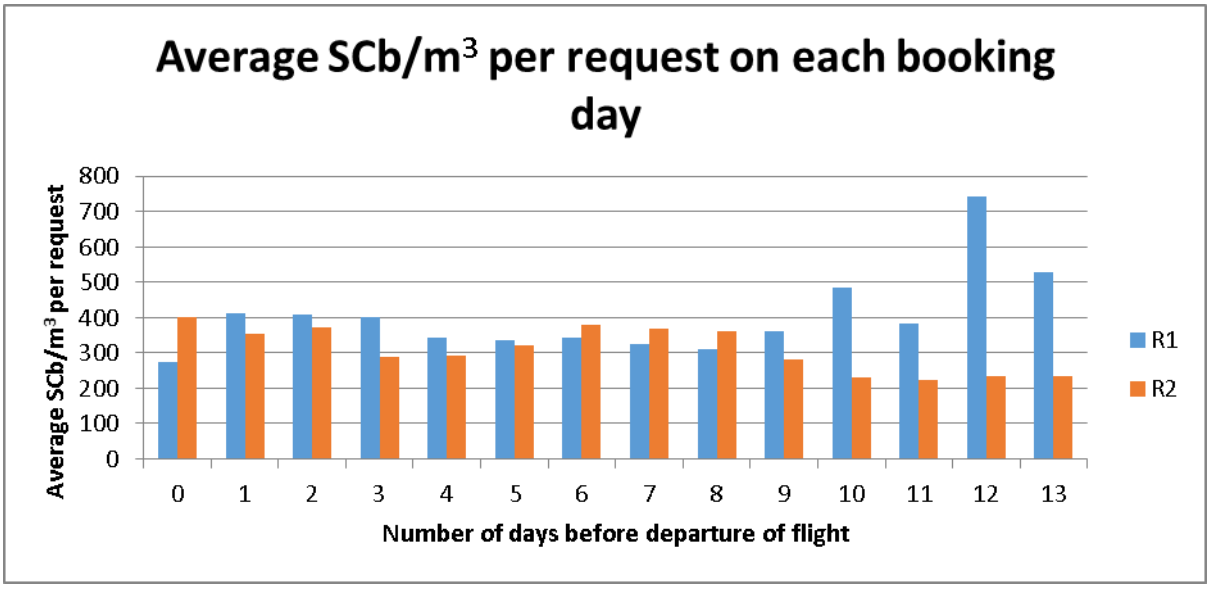


Figure 11: Average SCb/m^3 per request, for each route and amount of days before departure

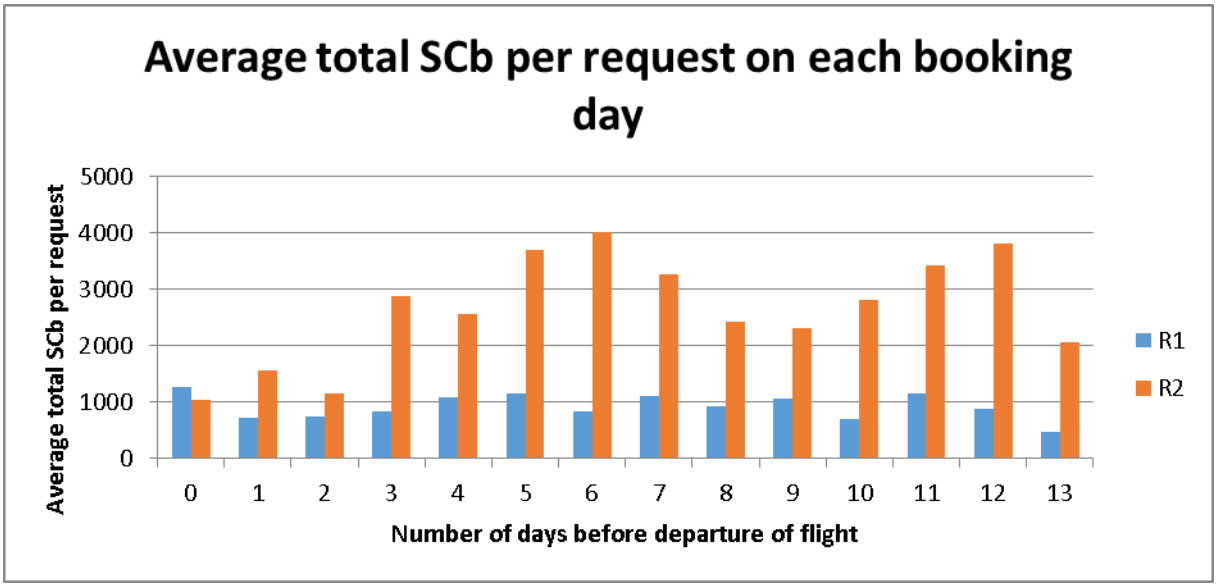


Figure 12: Average total SCb per request, for each route and amount of days before departure

Data on offloading costs is not used. Thus, the offloading function will be the zero function. This also allows the use of MP 4 in the dynamic programming algorithm, which considerably speeds up the algorithm. This means the running times in Table 3 are not representative for the further research. Also, data on pallet positions were not used. Finally, booking requests always involve just one flight and show-up rates were assumed to be 100%. Thus, the basic DP-algorithm without the two extensions of multiple flights and show-up rates can be used. The reason for above simplifications is the fact that this research is focused on the performance of various strategies for the booking problem, and above factors are theorized to only make the comparison more complex without changing the relative magnitude of the performance differences between strategies.

The following graphs shows the cumulative volume slope in green as a function of SCb per m^3 buckets of €25 each. These graphs show that the slope is increasing very fast in the lower buckets, meaning that increasing an EC by a little in these regions, will mean a relatively large amount of extra rejections, due to requests not achieving the EC.

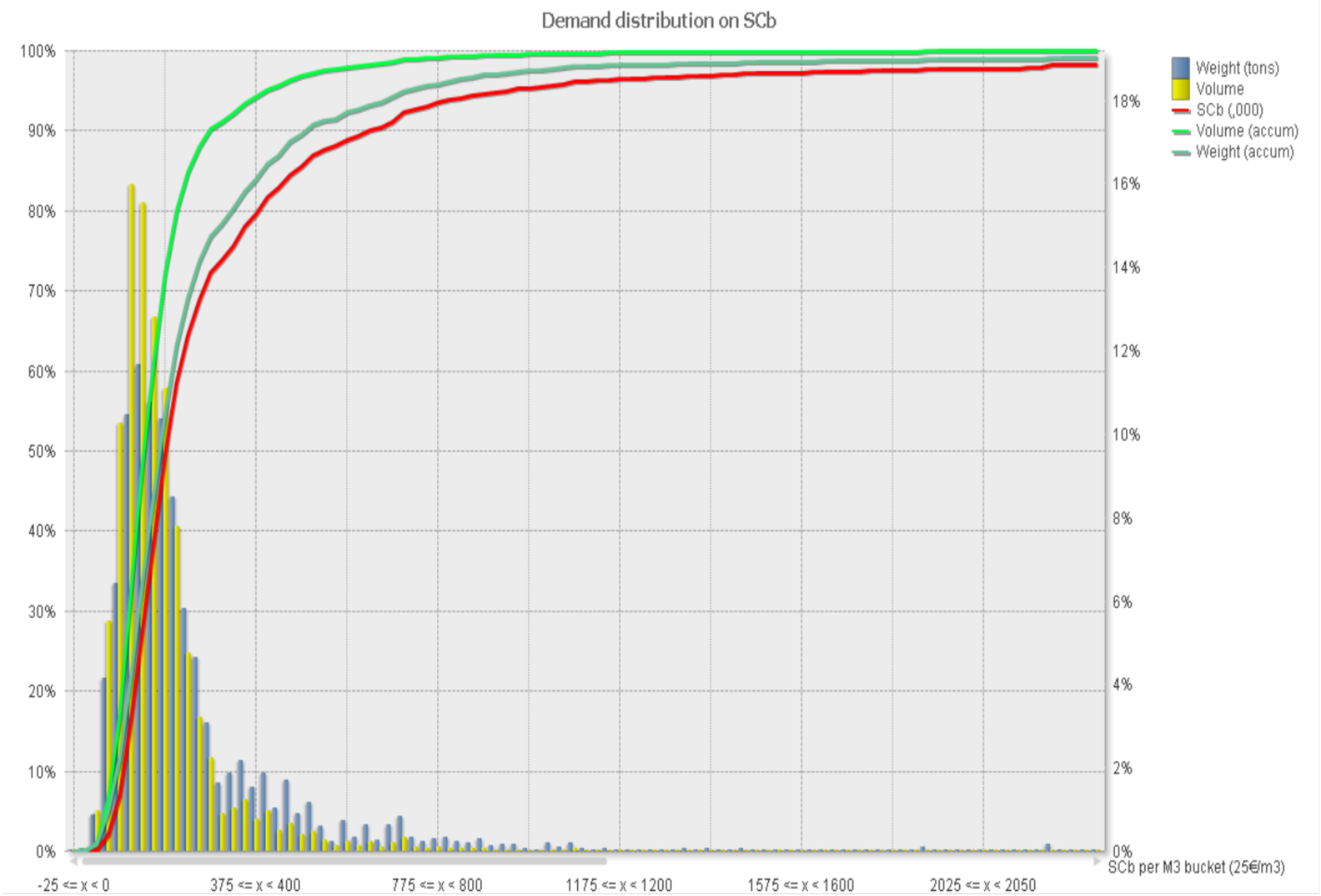


Figure 13: Volumes and weights per SCb/m^3 buckets, R1

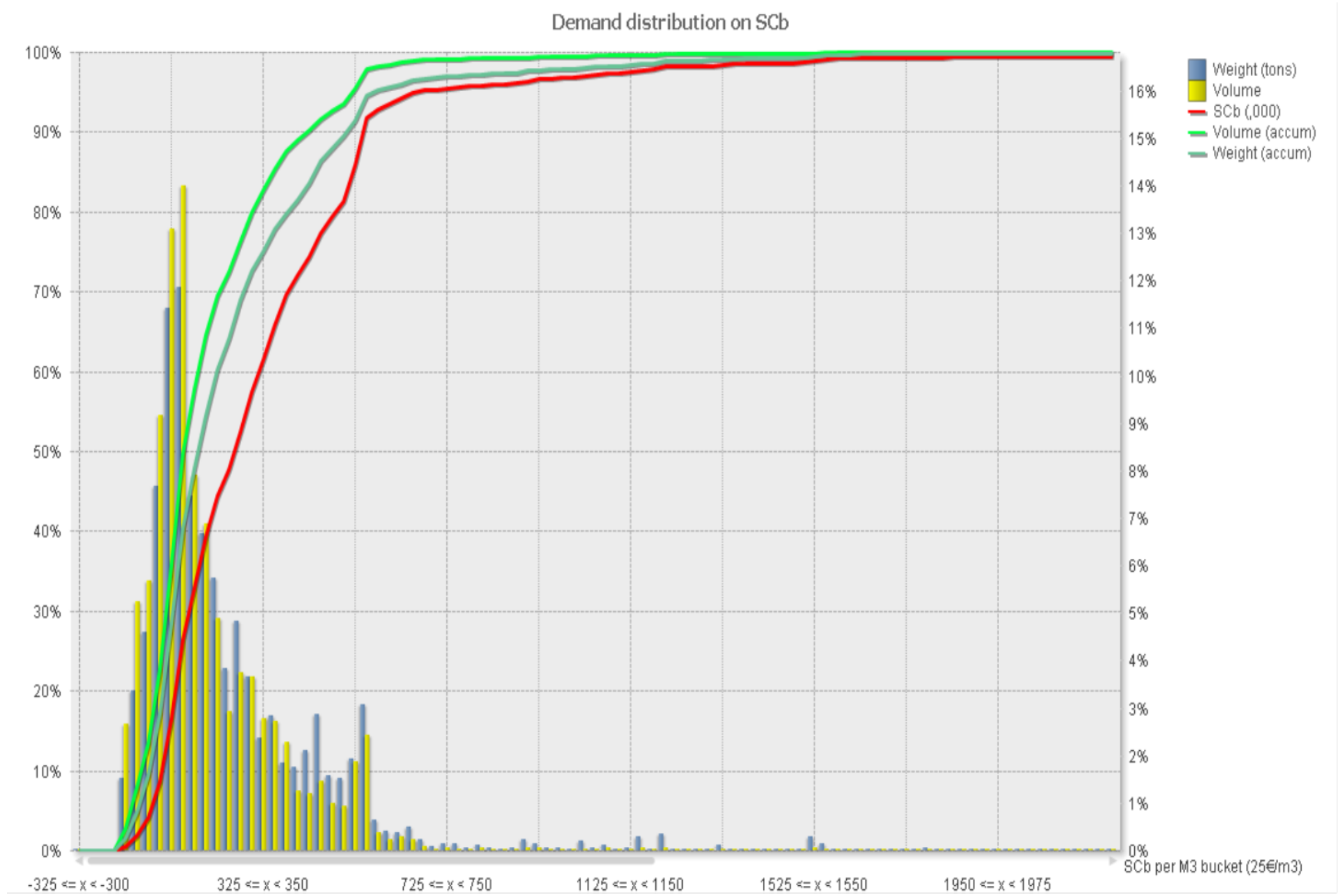


Figure 14: Volumes and weights per SCb/m^3 buckets, R2

In this chapter real-life data to be used in an investigation into the performance of the various strategies for the booking problem has been introduced and discussed in detail. In the next chapters, this investigation will be formalized and discussed.

CHAPTER 8 RESULTS OF BID-PRICE STRATEGIES OPTIMIZATION BASED ON SMALL INSTANCES

In this chapter, the real-life data discussed in the previous chapter will be used as input for an investigation into the performance of the various bid-price strategies discussed in Chapter 4, when the decision variables in these strategies are optimized. The decision variables in these strategies are the EC's and if applicable the bucket sizes. Because the formal mathematical models described in Chapter 5 are computationally quite heavy, the analysis in this chapter is based on rather small problem instances. This allows us to compare the various strategies in a fair manner, namely, in a manner where all strategies are executed in a nearly optimal way. However, as the number of scenarios used is small, the optimal decision variables found may not work well when used in other, larger scenario sets. To overcome this limitation of the analysis, in the next chapter heuristics to find decision variables that work well when using a very large number of scenarios will be explored.
This chapter and the next chapter have differing goals. The goal of this chapter is to determine the improvement potential when moving from simple bid-price strategies, for instance s1a, to more complex bid-price strategies, for instance s2b. Then, in the next chapter, we attempt to find well-working decision variables that hopefully are able to fulfill some of this improvement potential.

Solving times strategy 1a:

First, using T = 100 and N = 15, the performance of the latest versions of various well-known MIP-solvers will be tested on two generated scenario sets of 15 scenarios each, on route 1 and for various starting capacities. This implies a decision problem with nearly 20,000 constraints and around 13,500 variables of which almost 11,000 integer. The scenario sets are generated using the empirical distributions discussed in the previous chapter. The problems are solved to optimality. Capacities-setting C1 refers to a case of $\frac{T}{2}$ volume and T*1000 weight (volume-constrained flight), C2 refers to a case of $\frac{T}{2}$ volume and (weighted average density)* $\frac{T}{2}$ weight (sometimes volume constrained, sometimes weight-constrained, called a 'flip-flop' flight), C3 refers to a case of $\frac{T}{2}$ volume and T*40 weight (weight-constrained flight).

Table 6: Solving times for strategy 1a compared over various solvers

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
CPLEX 12.5	7.82 sec	13.26 sec	16.91 sec	9.63 sec	26.97 sec	15.38 sec
GUROBI 5.5	132.57 sec	182.52	154.24 sec	100.62 sec	146.14 sec	152.93 sec
MOSEK 6.0	>500 sec (terminated, 4.4% optimality gap)	>500 sec (terminated, 6% optimality gap)	-	-	-	-

Solving times strategy 1b:

The same settings as above are used on the two best performing solvers, except that an optimality gap of 2% is allowed, i.e. the solvers run until a feasible solution is found that is proven to be at most 2% worse than the optimal solution.

Table 7: Solving times for strategy 1b compared over two best performing solvers

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
CPLEX 12.5	414 sec (1.3% gap)	>500 sec	247 sec (1.9% gap)	>500 sec	289 sec (1.8% gap)	>500 sec
GUROBI 5.5	19.5 sec (1.8% gap)	213 sec (2% gap)	9.11 sec(1.5% gap)	6.99 sec (1.8% gap)	169 sec (1.9% gap)	33 sec (1.9% gap)

Solving times strategy 2a:

T = 20, *N* =10 is used, an optimality gap of 20% is allowed.

Table 8: Solving times for strategy 2a compared over two best performing solvers

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
CPLEX 12.5	242 sec (19.3% gap)	243 sec (16.8% gap)	>500 sec (48.1% gap)	>500 sec (23.1% gap)	>500 sec (37.3% gap)	>500 sec (36.9% gap)
GUROBI 5.5	212 sec (17.6% gap)	329 sec (15.7% gap)	>500 sec (76.2% gap)	>500 sec (30.3% gap)	447 sec (16.3% gap)	>500 sec (52.3% gap)

Solving times strategy 2b:

T = 20, *N* =10 is used, an optimality gap of 20% is allowed.

Table 9: Solving times for strategy 2b compared over two best performing solvers

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
CPLEX 12.5	102.9 sec (19.2% gap)	>500 sec (167.5% gap)	>500 sec (30.3% gap)	>500 sec (126.8% gap)	>500 sec (25.5% gap)	>500 sec (39.5% gap)
GUROBI 5.5	316.4 sec (19.9% gap)	199 sec (18.1% gap)	>500 sec (28.0% gap)	39 sec (18.4% gap)	218 sec (19.5% gap)	456 sec (19.7% gap)

For convenience, for the rest of thesis only one solver will be used. Although CPLEX performs much better than GUROBI in many cases, GUROBI will be used as the solver, because it performs better in the case of the most difficult problem (s2b), in which CPLEX performed very badly in some scenarios.

Comparison objective value between strategies 1a and 1b:

The optimal objective value of the following three approaches will be compared:

1) The optimal objective value corresponding to strategy 1a,single EC.
2) The optimal objective value of the MIP corresponding to strategy 1b, EC dependent on time.
3) The objective value of the MIP corresponding to the two-dimensional knapsack problem.

The scenario sets used will be held constant over the three problems. Thus we solve for a certain scenario set the knapsack problem for all scenarios individually and take as optimal objective value the average of the individual optimal objective values. Then, for the same scenario set, we use the MIP to find the optimal static entry condition or optimal time-dependent entry conditions.

As discussed before, the second objective value cannot be lower than the first objective value as the optimal solution of the first MIP is also feasible for the second MIP and they have the same objective function. It should also be clear that the third objective value cannot be worse than the second objective value, by the same reasoning. In fact, the objective value of the third MIP is the upper bound for the objective value of any accept/reject strategy.

T = 100 and N=15 will be used. In the MIP corresponding to strategy 1b an optimal EC is determined for every time period

Table 10: Comparison optimal solutions s1a, s1b and knapsack for R1

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
Single EC (s1a)	18809	14713	8022	18669	15657	8213
Time-dependent EC (s1b)	19524 - 19787[3]	17589 - 17937	10149 - 10269	19922 - 20246	18046 - 18344	9988 - 10155
Knapsack solution (upper bound)	19801	17980	10255	20238	18373	10247

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
Single EC (s1a), difference with optimum	5%	18.2%	21.8%	7.8%	17.3%	19.8%
Time-dependent EC (s1b), difference with optimum	$\leq$ 1.4%	$\leq$2.2%	$\leq$1%	$\leq$1.6%	$\leq$1.8%	$\leq$2.5%
Time-dependent EC (s1b), improvement compared to s1a	$\geq$ 3.8%	$\geq$ 19.5%	$\geq$ 26.5%	$\geq$ 6.7%	$\geq$ 15.3%	$\geq$ 21.6%

[3] First value corresponds to best-found solution, upper value is best-found upper bound.

Table 11: Comparison optimal solutions s1a, s1b and knapsack for R2

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
Single EC (s1a)	22920	18776	11006	22897	20199	13138
Time-dependent EC (s1b)	25224 -25303	21855 - 22181	17401 - 17631	29113 - 29176	22000 - 22236	16776 - 17043
Knapsack solution (upper bound)	25413	22262	18094	29335	22411	17530

	C1, S1	C2, S1	C3, S1	C1, S2	C2, S2	C3, S2
Single EC (s1a), difference with optimum	9.8%	15.6%	39.2%	21.9%	9.9%	25%
Time-dependent EC (s1b), difference with optimum	≤0.74%	≤1.83%	≤3.83%	≤0.76%	≤1.9%	≤4.31%
Time-dependent EC (s1b), improvement compared to s1a	≥ 10%	≥ 16.3%	≥ 58.1%	≥ 27.1%	≥ 8.9%	≥ 27.6%

Sub conclusions:

The results so far from this chapter are combined in the following table:

Table 12: Sub conclusions Chapter 8

	Average improvement from s1a to s1b, MIP	Average closure of optimality gap between knapsack and s1a, by using s1b, MIP
R1, C1	5.3%	76.8%
R1, C2	17.3%	88.0%
R1, C3	24.0%	91.4%
R2, C1	18.6%	95.4%
R2, C2	12.5%	85.6%
R2, C3	41.5%	86.4%
C1, average over two routes	12.0%	86.1%
C2, average over two routes	14.9%	86.8%
C3, average over two routes	32.8%	88.9%

Using time-dependent EC's instead of static EC's led to an average improvement of over 10% in the MIP, and more significantly and more importantly, given that there simply is an upper bound on the possible increase, closes the optimality gap by more than 86% on average. This shows that the strategy at least has some theoretical merit. As expected, the improvements gained were larger in the cases of C2 and C3, where in practice dual bid-prices or an EC on SCb/kg would be used. However, interestingly enough, the closure of the optimality gap is also very large in cases C2 and C3, showing that a strategy of using time-dependent EC's could possibly diminish the added value of using dual bid-prices at all.

Comparison of all bid-price strategies:

Now, strategies 2a and 2b will be added to the comparison. The various strategies are tested using small scenario sets of 10 with 20 requests each. In strategies 2a and 2b, three buckets are used. The algorithms were allowed 1000 seconds of solving time. In the case of R1, C1, S1, strategy s2a was solved to optimality. The average is taken over the best found feasible

solutions. Also added is the achieved cumulative SCb when using a first-come first-served (FCFS) strategy, i.e. s1a with EC forced at 0.

It is straightforward to extend MP 2 to also include an EC based on weight. The implementation used is to accept a request when it fits and it satisfies *both* EC's: the EC on volume and the EC on weight. Note that variations are possible and this may not be the optimal implementation. Including a second EC is interesting in the case of C2, flip-flop flights and C3, weight-constrained flights. Obviously, including an EC based on weight can never decrease the optimal objective function for a given constraint setting and scenario set, as the EC on weight can simply be set to 0, in which case it has no effect.

Table 13: Comparison of optimal objective values for various bid-price strategies, R1, C1

R1, C1	S1	S2	S3	S4	S5	Average
FCFS	2712	2594	2472	2423	2585	2519
S1a	3155	3414	3029	2897	3521	3203
S1b	3235	3462	3198	2998	3672	3313
S2a	3226	3468 - 3658	3112 - 3378	3051 - 3219	3538 - 3814	3279
S2b	3274 – 3450	3468 - 3663	3198 - 3484	3090- 3289	3758- 3870	3358
Knapsack	3396	3488	3396	3251	3780	3462

Table 14: Comparison of optimal objective values for various bid-price strategies, R1, C2

R1, C2	S1	S2	S3	S4	S5	Average
FCFS	2234	2194	2623	2423	2441	2383
S1a	2527	2844	2863	2603	2971	2762
S1a (dual EC)	2983	3049	3005	2682	3068	2951
S1b	2840	3060	3049	2718	3162	2966
S2a	2710 - 2869	2883 – 3208	2966 - 3176	2730 - 3011	3060 – 3376	2870
S2b	3009 - 3314	3150 – 3382	3134 - 3373	2742 – 3040	3190 - 3437	3045
Knapsack	3130	3197	3273	2981	3296	3175

Table 15: Comparison of optimal objective values for various bid-price strategies, R1, C3

R1, C3	S1	S2	S3	S4	S5	Average
FCFS	1368	1126	1525	1286	1171	1295
S1a (EC on volume)	1702	1743	1631	1510	1820	1681
S1a (EC on weight)	1702	1746	1701	1532	1848	1707
S1b	1707	1778	1682	1548	1865	1716
S2a	1702	1743	1643	1523	1820	1686
S2b	1749	1781 - 1833	1703	1548	1865	1729
Knapsack	1795	1830	1812	1594	1895	1785

Table 16: Comparison of optimal objective values for various bid-price strategies, R2, C1

R2, C1	S1	S2	S3	S4	S5	Average
FCFS	2975	2614	3010	2737	3005	2868
S1a	3355	3377	3454	3756	3700	3572
S1b	3570	3570	3617	3858	4734	3870
S2a	3538 - 3756	3538 - 3917	3583 - 3885	3841 - 4088	4128	3726
S2b	3579 - 3896	3682 - 3948	3619 - 3982	3914 - 4097	4991 - 5285	3957
Knapsack	3890	3848	3835	3999	5245	4163

Table 17: Comparison of optimal objective values for various bid-price strategies, R2, C2

R2, C2	S1	S2	S3	S4	S5	Average
FCFS	2640	2323	2228	2392	2548	2426
S1a	3144	2847	2802	2916	2864	2915
S1a (dual EC)	3219	2980	3170	3113	3144	3125
S1b	3280	3043	3149	3205	3094	3154
S2a	3173	2959	2966	3104	3132	3067
S2b	3340 – 3750	3092 – 3429	3249 – 3598	3144 – 3564	3150 – 3643	3195
Knapsack	3518	3289	3490	3350	3479	3425

Table 18: Comparison of optimal objective values for various bid-price strategies, R2, C3

R2, C3	S1	S2	S3	S4	S5	Average
FCFS	1718	1633	1536	1624	1842	1671
S1a (volume)	2129	2055	2175	2218	2184	2152
S1a (weight)	2269	2218	2444	2393	2258	2316
S1b	2282	2337	2433	2303	2466	2364
S2a	2182	2158	2292	2277	2330	2248
S2b	2335 - 2501	2412	2467 - 2568	2347 - 2547	2610 - 2775	2434
Knapsack	2546	2539	2757	2592	2749	2659

Conclusions:

Table 19: Conclusions Chapter 8

	Average improvement from s1a to s1b, MIP	Average closure of optimality gap between knapsack and s1a, by using s1b, MIP	Average improvement from s1a to s2a, MIP	Average closure of optimality gap between knapsack and s1a, by using s2a, MIP	Average improvement from s1a to s2b, MIP	Average closure of optimality gap between knapsack and s1a, by using s2b, MIP
C1, average over two routes	5.9%	46.5%	3.4%	27.7%	7.8%	62.5%
C2, average over two routes	7.8%	48.2%	5.7%	43.2%	11%	64%
C3, average over two routes	8.7%	37.8%	2.4%	23.3%	8%	50.9%

The improvement potential of s1b over s1a is confirmed again, although the improvement is smaller this time. This might be due to the number of request having decreased. S2a also offered improvements in objective function over s1a, albeit smaller than in the case of s1b. S2b combines the improvement potential of both s1b and s2a, in the cases of C1and C2 even

exceeding the sum of the two individual increases. It is also interesting to see that s1b with EC's based on volume outperformed a dual bid-price strategy even in flip-flop and volume constrained settings. This can indicate that the whole process of determining which EC to use may be obsolete in real-life cases such as those in air cargo, with a relatively constant density of requests.

Having ascertained the improvement potential of more complex bid-price strategies over static bid-price strategies in small instances, in the next chapter it will be analyzed whether this improvement potential can also be lived up to in larger instances, using heuristics to find well-performing decision variables.

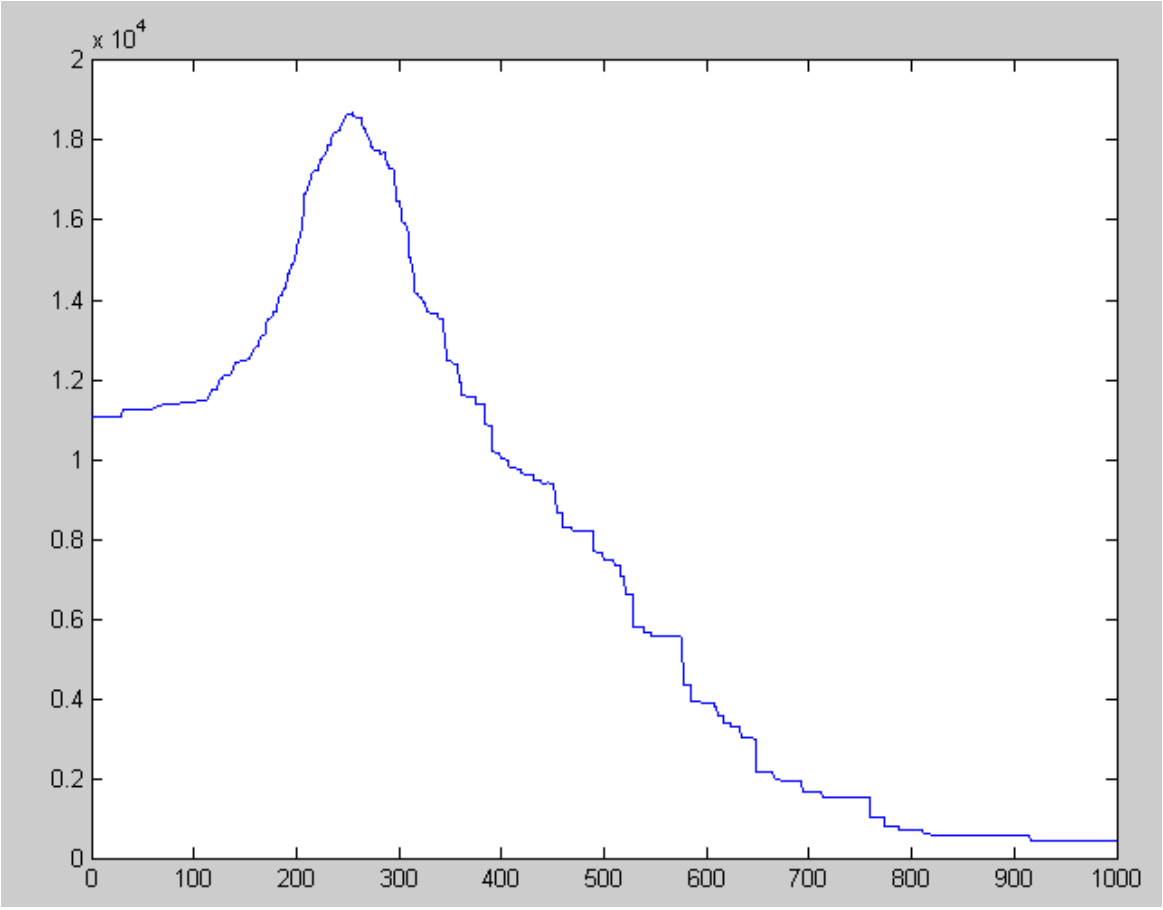
CHAPTER 9 COMPARISON OF PERFORMANCE OF BID-PRICE STRATEGIES USING HEURISTICS IN LARGE INSTANCES

In this chapter, the developed simulation will be used to try and find well-performing decision variables for the various bid-price strategies. As the simulation is much less complex than the MIPs used, much more scenarios can be used in this approach.

Comparison strategy 1b vs. strategy 1a using heuristics:

The following approach will be used to assess the practical effect of strategy 1b vis-à-vis strategy 1a. The optimal EC for strategy 1a will be determined by simulation: the cumulative SCb for various ECs will be determined over 100 runs. EC will be initialized at 0 and incremented by 1 until the EC has reached the level of the maximum ratio between SCb and volume in the data set used. The optimal EC is the EC with the highest average cumulative SCb. Note that this is not necessarily optimal but likely to be close to the real optimum. This generates the following graph of average cumulative SCb as a function of EC in the case of R1, C1. The optimal simulated EC equals 247.

Figure 15: Example of average cum. SCb as function of EC



Strategy 1b will be implemented with five different ECs, i.e. there is not an EC for each time period individually but *T* will be divided in five distinct subperiods with an EC for each. The MIP for strategy 1b will be solved three times for three scenario sets of 34 scenarios with a tolerated optimality gap of 2%. This has three solution sets with five ECs as a result. For each of the five ECs the 'optimal' EC taken is the median of the three ECs found for that period.

Table 20: Obtained strategies for s1b

Strategy 1b ECs R1, C1	Scenario set 1	Scenario set 2	Scenario set 3	Median
EC1 (valid for requests 1 to 20)	285.1372661	310.1471897	306.6042052	306.6042052
EC2 (valid for request 21 to 40)	301.0354213	268.0882676	289.9380762	289.9380762
EC3 (valid for requests 41 to 60)	271.948127	253.5155126	289.9380762	271.948127
EC4 (valid for requests 61 to 80)	262.2381668	263.3953301	238.4410661	262.2381668
EC5 (valid for requests 81 to 100)	218.4307554	193.9025328	181.7130546	193.9025328

In order to compare the simulated average cumulative SCb between the best obtained variable setting for strategy 1a (i.e. with EC = 247) and the best obtained variable setting for strategy 1b, i.e. the median values in above table, these ECs were used in the simulation with 100,000 runs.

Table 21: Differences in objective function between strategies 1a and 1b in detail

R1 C1	Strategy 1a	Strategy 1b
Average cumulative SCb	18581	19171 (3.17% improvement, as opposed to 3.8% theoretically)
Standard deviation cumulative SCb	2541	2873
Coefficient of variation	13.7%	15%
Maximum cumulative SCb	35221	35300
Minimum cumulative SCb	6429	7549
Median cumulative SCb	18276	18723
25th percentile cumulative SCb	17184	17427
75th percentile cumulative SCb	19585	20341

The same analysis will be repeated for capacity settings C2 and C3.

In the case of C2, the median of the optimal EC's for the five periods over three scenario sets were found to be (305.1703229, 294.904859, 285.1372661, 253.5155126, 198.3727831). The best found static EC was 250. Simulating 100.000 runs using this static EC-setting yields average cumulative SCb of 14756 (standard deviation of 2139). Over 100.000 runs, using the found time-dependent EC's yields average cumulative SCb of 16160 (standard deviation of 2051), an increase of 9.5%.

In the case of C3, the median of the optimal EC's for the five periods over three scenario sets were found to be (350.233361, 345.065449, 318.784254, 302.472625, 227.06098). The best found static EC was 288. Simulating 100.000 runs using this static EC-setting yields average cumulative SCb of 8158 (standard deviation of 1651). Over 100.000 runs, using the found time-dependent EC's yields average cumulative SCb of 9102 (standard deviation of 1631), an increase of 11.6%.

R2:

In the case of C1, the median of the optimal EC's for the five periods over three scenario sets were found to be (450.140898, 382.558424, 355.222900, 435.163999, 363.120523). The best found static EC was 388. Simulating 100.000 runs using this single EC-setting yields average

cumulative SCb of 23581 (standard deviation of 4743). Over 100.000 runs, using the found time-dependent EC's yields average cumulative SCb of 25031 (standard deviation of 4897), an increase of 6.1%.

In the case of C2, the median of the optimal EC's for the five periods over three scenario sets were found to be (400.48425, 390.8327, 341.25834, 460.23526, 310.52362). The best found static EC was 345. Simulating 100.000 runs using this static EC-setting yields average cumulative SCb of 18615 (standard deviation of 4273). Over 100.000 runs, using the found time-dependent EC's yields average cumulative SCb of 19969 (standard deviation of 4455), an increase of 7.3%.

In the case of C3, the median of the optimal EC's for the five periods over three scenario sets were found to be (390.8327, 363.120523, 338.52943, 401.2353, 350.21424). The best found static EC was 366. Simulating 100.000 runs using this static EC-setting yields average cumulative SCb of 12535 (standard deviation of 4633). Over 100.000 runs, using the found time-dependent EC's yields average cumulative SCb of 14310 (standard deviation of 4973), an increase of 14.2%.

Sub conclusions:

We extend Table 12 to include the results from the results from the applied heuristic in the last column:

Table 22: Comparison between MIP improvement and improvement in practice

	Average improvement from s1a to s1b, MIP	Average closure of optimality gap between knapsack and s1a, by using s1b, MIP	Average improvement from s1a to s1b, in practice
R1, C1	5.3%	76.8%	3.2%
R1, C2	17.3%	88.0%	9.5%
R1, C3	24.0%	91.4%	11.6%
R2, C1	18.6%	95.4%	6.1%
R2, C2	12.5%	85.6%	7.3%
R2, C3	41.5%	86.4%	14.2%
C1, average over two routes	12.0%	86.1%	4.7%
C2, average over two routes	14.9%	86.8%	8.4%
C3, average over two routes	32.8%	88.9%	12.9%

It can be seen that some of the improvement potential obtained in the analysis using MIPs was fulfilled in practice. However, the improvement is smaller in the case of using heuristics and large instances than in the case of using MIPs and small instances. This might show that the used heuristic for strategy 1b is not sophisticated enough or not enough input data was used. Remember that median EC's were taken over only three scenario sets. Still, it has been

shown that it is quite easy to improve on a static bid-price strategy using time-dependent bid-prices.

Comparison strategy 2a vs. strategies 1a and 1b using heuristics:

First of all, it was attempted to use the found solutions for the MIP in the previous chapter. The found solutions were the following (R1, C1):

Table 23: Example of optimal decision variables for strategy 2a, R1, C1

	S1	S2	S3	S4	S5
Bucket proportion 1	0.39	0	0.68	0.20	0.34
Bucket proportion 2	0.26	0.24	0.24	0.15	0.53
Bucket proportion 3	0.35	0.76	0.08	0.65	0.13
EC_1	203.108680	170.0093701	203.814541	144.8420925	232.8626728
EC_2	206.7854826	182.8760386	241.2229025	242.6570288	233.4502628
EC_3	241.2229025	243.1817595	488.3872046	260.3676805	452.6339378

Simulating the found strategies from table 14 over 10.000 runs, results in the following average cumulative SCb:

Table 24: Average cum. SCb over 10.000 runs for found strategies, s2a

	S1	S2	S3	S4	S5
Average cumulative SCb	3078	2993	3109	2879	3078

Optimal EC for s1a, based on 1000 runs, was found to be 209. Average cumulative SCb for this strategy (100.000 runs) = 3150, larger than any of the values in above table.

This shows that the amount of scenarios used in the MIP is unfortunately too small. However, as the MIP is too computationally demanding to handle more scenarios, this is not a viable option. It also shows that finding bucket sizes and entry conditions for each bucket that outperform strategy 1a is a difficult problem, as even strategies that make sense in certain cases, i.e. above strategies, do not work better than strategy 1a in a large scenario set. As this approach does not seem effective, it will not be attempted in the remainder of this thesis. Instead, well-working decision variables will be attempted to be found by use of a heuristic.

The proposed heuristic H for determining the optimal decision variables for strategy 2a, i.e. the various ECs and the bucket sizes, is the following:

Step 1: initialize with random bucket proportions adding to 1 (c_1, c_2, c_3) and all three ECs equal to the optimal EC from strategy 1a. Note that this strategy is exactly the same as strategy 1a with optimal EC.

Step 2: Try to improve the EC of the highest bucket (EC_3) by simulating over all values of this highest EC in the range of [EC_2, max(SCb /m^3)] by taking appropriate increments in EC_3. In iteration *i*, EC_3 is set to EC_{3i}. Next, determine the optimal EC_2 in the range of [EC_1, EC_{3i}]. In an equivalent manner, determine the optimal EC_1.

Step 3: Repeat step 2 until the relative improvement is smaller than a certain threshold value. We now have approximately optimal values of ECs for the current bucket proportions.

Step 4: Try to improve the bucket proportions for the current ECs by simulating over all allowed combinations of the bucket proportions, i.e. all proportions larger than or equal to 0 and adding up to 1, by taking appropriate increments in the bucket proportions.

Step 5: Repeat step 4 until the relative improvement is smaller than a certain threshold value. We now have approximately optimal values of bucket proportions for the current ECs.

Step 6: Repeat steps 1 to 5 any desired number of times.

Note that in each iteration of step 2 and step 4 the objective value cannot decrease, except due to not simulating enough runs. Also, after step 5 the heuristic will terminate in a local optimum, although this does not have to be a global optimum. This is why in step 6 we repeat a number of times, each time with different starting bucket proportions.

In practice, it is often observed that steps 4 and 5 are redundant and the local optimum is already obtained after step 3.
No heuristics have been attempted in order to find well-working decision variables for strategy 2b.

R1, C1:
Using the simulation for s1a, based on 1000 runs, the best EC was found to be 209. Average cumulative SCb for this strategy (100.000 runs) = 3150, standard deviation of 840.

Using a scenario set of 200 scenarios, best time-dependent EC's for the five periods were found to be (228.4082901, 233.4502628, 206.7854826, 212.8951503, 155.8775431) using the MIP for s1b. Simulating strategy 1b with this variable setting over 100.000 runs, an average cumulative SCb of 3187 is obtained with standard deviation of 818.

Heuristic H was attempted, which resulted in EC's of (206, 218, 236), with bucket proportions of (0.28, 0.51, 0.21), an average cumulative SCb of 3172 over 100.000 runs, with standard deviation of 837.

Table 25: Comparison of performance of various bid-price strategies in practical cases, R1, C1

R1, C1	S1a	S1b	S2a
Average cumulative SCb over 100.000 runs (standard deviation)	3150 (840)	3187 (818)	3172 (837)

R1, C2:
Using the simulation for s1a, based on 1000 runs, the best EC was found to be 209. Average cumulative SCb for this strategy (100.000 runs) = 2759, standard deviation of 519.

Using a scenario set of 200 scenarios, best time-dependent EC's for the five periods were found to be (307.0706762, 311.1660684, 206.7854826, 209.2647161, 180.2130421). Simulating strategy 1b with this variable setting over 100.000 runs, average cumulative SCb of 2798 is obtained (standard deviation of 510).

Heuristic H was attempted, which resulted in EC's of (185, 207, 220), with bucket proportions of (0.26, 0.47, 0.27), average cumulative SCb of 2783 over 100.000 runs, standard deviation of 476.

Table 26: Comparison of performance of various bid-price strategies in practical cases, R1, C2

R1, C2	S1a	S1b	S2a
Average cumulative SCb over 100.000 runs (standard deviation)	2759 (519)	2798 (510)	2783 (476)

R1, C3:

Using the simulation for s1a, based on 1000 runs, the best EC was found to be 209. Average cumulative SCb for this strategy (100.000 runs) = 1635, standard deviation of 384.

Using a scenario set of 200 scenarios, best time-dependent EC's for the five periods were found to be (268.0879678, 311.1660684, 311.1660684, 182.8760386, 209.2647161). Simulating strategy 1b with this variable setting over 100.000 runs, average cumulative SCb of 1655 is obtained (standard deviation of 391).

Heuristic H was attempted, which did not find any strategy that worked better than a single EC-strategy.

Table 27: Comparison of performance of various bid-price strategies in practical cases, R1, C3

R1, C3	S1a	S1b	S2a
Average cumulative SCb over 100.000 runs (standard deviation)	1635 (384)	1655 (391)	1635 (384)

R2, C1:

Using the simulation for s1a, based on 1000 runs, the best EC was found to be 281. Average cumulative SCb for this strategy (100.000 runs) = 3350, standard deviation of 974.

Using a scenario set of 200 scenarios, best time-dependent EC's for the five periods were found to be (322.6000806, 306.3608255, 282.1306833, 236.0780671, 175.8683446). Simulating strategy 1b with this variable setting over 100.000 runs, average cumulative SCb of 3418 is obtained (standard deviation of 914).

Heuristic H was attempted, which resulted in EC's of (236, 280, 280) with bucket proportions of (0.3, 0.35, 0.35), average cumulative SCb of 3370 over 100.000 runs, standard deviation of 900.

Table 28: Comparison of performance of various bid-price strategies in practical cases, R2, C1

R2, C1	S1a	S1b	S2a
Average cumulative SCb over 100.000 runs (standard deviation)	3350 (974)	3418 (914)	3370 (900)

R2, C2:

Using the simulation for s1a, based on 1000 runs, the best EC was found to be 237. Average cumulative SCb for this strategy (100.000 runs) = 2849, standard deviation of 672.

Using a scenario set of 200 scenarios, best time-dependent EC's for the five periods were found to be (322.6000806, 338.163215, 259.188750, 247.641577, 236.078067). Simulating strategy 1b with this variable setting over 100.000 runs, average cumulative SCb of 2884 is obtained (standard deviation of 680).

Heuristic H was attempted, which resulted in EC's of (230, 240, 290) with bucket proportions of (0.12, 0.67, 0.21), average cumulative SCb of 2854 over 100.000 runs, standard deviation of 686.

Table 29: Comparison of performance of various bid-price strategies in practical cases, R2, C2

R2, C2	S1a	S1b	S2a
Average cumulative SCb over 100.000 runs (standard deviation)	2849 (672)	2884 (680)	2854 (686)

R2, C3:

Using the simulation for s1a, based on 1000 runs, the best EC was found to be 273. Average cumulative SCb for this strategy (100.000 runs) = 2037, standard deviation of 736.

Using a scenario set of 200 scenarios, best time-dependent EC's for the five periods were found to be (322.60008, 341.814584, 306.360825, 296.435655, 236.078067). Simulating strategy 1b with this variable setting over 100.000 runs, average cumulative SCb of 2121 is obtained (standard deviation of 759).

Heuristic H was attempted, which resulted in EC's of (270, 270, 355) with bucket proportions of (0.10, 0.10, 0.9), average cumulative SCb of 2042 over 100.000 runs, standard deviation of 737.

Table 30: Comparison of performance of various bid-price strategies in practical cases, R2, C3

R2, C3	S1a	S1b	S2a
Average cumulative SCb over 100.000 runs (standard deviation)	2037 (736)	2121 (759)	2042 (737)

Conclusions:

We have seen that in all cases it was possible to find using a heuristic decision variables that improved expected cumulative SCb using s1b compared to s1a. Also, in all but one of the six cases it was possible to find using heuristic H decision variables that improved expected cum. SCb using s2a compared to s1a. In all of the six cases: the relationship $obj(s1b) > obj(s2a) \geq obj(s1a)$ holds. This is consistent with the results from earlier theoretical optimization using MIPs. We will extend Table 19 with the practical results from this chapter.

Table 31: Conclusions Chapter 9

	Average improve-ment from s1a to s1b, MIP	Average closure of optimality gap between knapsack and s1a, by using s1b, MIP	Average improve-ment from s1a to s1b, using in practice	Average improveme-nt from s1a to s2a, MIP	Average closure of optimality gap between knapsack and s1a, by using s2a, MIP	Average improve-ment from s1a to s2a, using heuristic in practice	Average improvem-ent from s1a to s2b, MIP	Average closure of optimality gap between knapsack and s1a, by using s2b, MIP
C1, average over two routes	5.9%	46.5%	1.6%	3.4%	27.7%	0.7%	7.8%	62.5%
C2, average over two routes	7.8%	48.2%	1.3%	5.7%	43.2%	0.6%	11%	64%
C3, average over two routes	8.7%	37.8%	2.7%	2.4%	23.3%	0.2%	8%	50.9%

It is interesting to see that although average improvements are larger in comparing the MIP solutions than in comparing the heuristic solutions, in both the MIPs as the heuristic solutions, s1b performs better than s2a. Thus, the results from the two analyses are consistent. It would seem that s1b is stronger from a theoretical optimization standpoint than s1a, as well as it is easier to find well-working decision variables for s1b than for s2a.

Having analyzed the performance of the various bid-price strategies in depth using both MIPs for small instances and heuristics for large instances, the next analysis is to analyze the performance of the developed DP-algorithm.

CHAPTER 10 THEORETICAL PERFORMANCE OF THE DP-ALGORITHM

First of all, the performance of the DP-algorithm will be tested in the same scenario sets as before, using 200 prediction scenarios.

Table 32: Comparison of DP-algorithm objective function with objective function of bid-price strategies, R1

R1, C1	S1	S2	S3	S4	S5	Average
FCFS	2712	2594	2472	2423	2585	2519
S1a	3155	3414	3029	2897	3521	3203
S1b	3235	3462	3198	2998	3672	3313
S2a	3226	3468 - 3658	3112 - 3378	3051 - 3219	3538 - 3814	3279
S2b	3274 – 3450	3468 - 3663	3198 - 3484	3090- 3289	3758- 3870	3358
DP	3179	3235	3021	2778	3535	3150
Knapsack	3396	3488	3396	3251	3780	3462

R1, C2	S1	S2	S3	S4	S5	Average
FCFS	2234	2194	2623	2423	2441	2383
S1a (volume)	2527	2844	2863	2603	2971	2762
S1a (dual EC)	2983	3049	3005	2682	3068	2951
S1b	2840	3060	3049	2718	3162	2966
S2a	2710 - 2869	2883 – 3208	2966 - 3176	2730 - 3011	3060 – 3376	2870
S2b	3009 - 3314	3150 – 3382	3134 - 3373	2742 – 3040	3190 - 3437	3045
DP	2842	2945	3016	2663	2952	2884
Knapsack	3130	3197	3273	2981	3296	3175

R1, C3	S1	S2	S3	S4	S5	Average
FCFS	1368	1126	1525	1286	1171	1295
S1a (volume)	1702	1743	1631	1510	1820	1681
S1a (weight)	1702	1746	1701	1532	1848	1707
S1b	1707	1778	1682	1548	1865	1716
S2a	1702	1743	1643	1523	1820	1686
S2b	1749	1781 - 1833	1703	1548	1865	1729
DP	1689	1743	1745	1505	1820	1700
Knapsack	1795	1830	1812	1594	1895	1785

Table 33: Comparison of DP-algorithm objective function with objective function of bid-price strategies, R2

R2, C1	S1	S2	S3	S4	S5	Average
FCFS	2975	2614	3010	2737	3005	2868
S1a	3355	3377	3454	3756	3700	3572
S1b	3570	3570	3617	3858	4734	3870
S2a	3538 - 3756	3538 - 3917	3583 - 3885	3841 - 4088	4128	3726
S2b	3579 - 3896	3682 - 3948	3619 - 3982	3914 - 4097	4991 - 5285	3957
DP	3153	3277	3189	3376	3423	3316
Knapsack	3890	3848	3835	3999	5245	4163

R2, C2	S1	S2	S3	S4	S5	Average
FCFS	2640	2323	2228	2392	2548	2426
S1a (volume)	3144	2847	2802	2916	2864	2915
S1a (dual EC)	3219	2980	3170	3113	3144	3125
S1b	3280	3043	3149	3205	3094	3154
S2a	3173	2959	2966	3104	3132	3067
S2b	3340 – 3750	3092 – 3429	3249 – 3598	3144 – 3564	3150 – 3643	3195
DP	3130	2890	2980	3050	3132	3013
Knapsack	3518	3289	3490	3350	3479	3425

R2, C3	S1	S2	S3	S4	S5	Average
FCFS	1718	1633	1536	1624	1842	1671
S1a (volume)	2129	2055	2175	2218	2184	2152
S1a (weight)	2269	2218	2444	2393	2258	2316
S1b	2282	2337	2433	2303	2466	2364
S2a	2182	2158	2292	2277	2330	2248
S2b	2335 - 2501	2412	2467 - 2568	2347 - 2547	2610 - 2775	2434
DP	1983	2134	2418	2141	1936	2122
Knapsack	2546	2539	2757	2592	2749	2659

Table 34: Conclusions Chapter 10

	Difference of DP-algorithm compared to s1a (volume), MIP	Difference of DP-algorithm compared to s1a (dual EC), MIP
C1, average over two routes	-4.4%	-4.4%
C2, average over two routes	+3.9%	-2.9%
C3, average over two routes	-0.1%	-4.4%

The DP-algorithm seems to have the most potential in a flip-flop-setting. It should be noted that the MIP objective value for s1a might be a bit optimistic because the EC is optimized in a clairvoyant manner based on a scenario set of only ten scenarios. In contrast, the DP-algorithm is non-clairvoyant. Thus, it might be expected that s1a will perform worse in a practical setting, whereas the DP-algorithm will not. This is the subject of the following chapter.

CHAPTER 11 NUMERICAL RESULTS FOR THE DP-ALGORITHM IN PRACTICE

First of all, the effect of the number of prediction scenarios will be tested. One might assume that a larger number of prediction scenarios will lead to a more sound result. However, taking for instance two times as much prediction scenarios, will also increase the computation time by approximately a factor 2. Thus, there is some tradeoff between this benefit and downside.

The R1 data will be used for testing the effect with T=20 and using capacities C3, i.e. a weight-constrained flight. The effect of the number of prediction scenarios will be tested over 100 simulation runs. Thus, there are 100 instances to be solved, and these 100 instances have been fixed, such that the difference in objective values between the various settings is purely due to the strategy used and not due to differences in the profitability of the instance. In the DP-algorithm, we use GUROBI to solve the knapsack problems. As we do not have an offloading function, the model is linear.

Table 35: Effect of number of prediction scenarios on DP-algorithm performance

	1 prediction scenario	2 prediction scenarios	5 prediction scenarios	10 prediction scenarios	50 prediction scenarios	200 prediction scenarios	500 prediction scenarios
Average cum. SCb over 100 runs	1545	1630	1687	1695	1706	1703	1706
Standard deviation	454	401	404	404	383	392	384

Note: the average cum. SCb for these 100 instances when solving the respective knapsack MIPs equals 1782.

Using the best static EC of 209 from the previous section yields average cumulative SCb of 1683 (standard deviation 384) over these 100 instances. Remember that s2a also attained this exact average cumulative SCb. Using the proposed strategies for s1b yielded 1693 (standard deviation 380). As can be seen in the above table, using 10 scenarios already performs about the same as the time-dependent EC setting strategy. Using more scenarios makes the DP-algorithm perform better than the time-dependent EC setting strategy, which was until now consistently the best performing of the strategies for which finding practical

solutions was feasible, i.e. excluding s2b for which we don't have a useful heuristic that finds the appropriate strategy variable settings. Note that using 50 scenarios or more is not time-consuming at all for solving one instance. Solving one instance using 50 decision scenarios takes approximately 3 seconds, and note that in real-life, one would not be solving an entire instance at once, but making the accept/reject decision for a single request per moment of request, which would take even considerably less time.

Generating a set of 1000 scenarios and using the DP-algorithm on each of these scenarios, using 100 prediction scenarios, the following average cumulative SCb's are achieved. Note that these values will be used for comparing the DP's performance against the heuristics used for determining decision variables in the cases of s1a, s1b and s2a.

Table 36: Average cum. SCb of heuristics

Setting used	DP-algorithm	S1a (EC on volume)	S1a (dual EC)	S1b	S2a
R1, C1	3175	3150	3150	3187	3172
R1, C2	2880	2759	2807	2798	2783
R1, C3	1660	1635	1649	1655	1635
R2, C1	3301	3350	3350	3418	3370
R2, C2	2945	2849	2890	2884	2854
R3, C3	2089	2037	2103	2121	2045

In the previous table, for s1a with dual EC, the following ECs were used in the case of C2:

Table 37: Found optimal solutions s1a with dual bid-prices

R1, C2	S1	S2	S3	S4	S5	Median
EC, volume	203.10868	206.785482	203.81454	200.309454	169.79182	203.1087
EC, weight	0.86139453	0.86734865	0.93073148	0.953310196	0.895792134	0.895792

R2, C2	S1	S2	S3	S4	S5	Median
EC, volume	199.084071	202.40183	190.847394	219.224711	203.554178	202.40183
EC, weight	1.37754403	1.3495952	1.39133961	1.39412148	1.54869963	1.39133961

Interestingly enough the optimal decision variables are quite constant over the scenario sets, which gives some hope that the median values of the decision variables might work quite well in a heuristic approach.

As the heuristic solution for s1a with dual bid-prices, the above median EC's were be used.

Table 38: Conclusions Chapter 11

	Difference of DP-algorithm compared to s1a (EC on volume), MIP	Difference of DP-algorithm compared to s1a (dual EC), MIP	Difference of DP-algorithm compared to s1a (EC on volume), heuristic	Difference of DP-algorithm compared to s1a (dual EC), heuristic	Difference of DP-algorithm compared to s1b, heuristic	Difference of DP-algorithm compared to s2a, heuristic
C1, average over two routes	-4.4%	-4.4%	-0.3%	-0.3%	-1.9%	-1.0%
C2, average over two routes	+3.9%	-2.9%	+3.9%	+2.3%	+2.5%	+3.4%
C3, average over two routes	-0.1%	-4.4%	+2.0%	+0.0%	-0.6%	+1.8%

As can be seen from Table 38, although the DP-algorithm performed worse than strategy 1b in small instances using MIPs, it performed very well in large instances where the bid-price strategies' performance dropped because of the difficulty of finding the optimal decision variables. In the constraint settings of C1 and C3, which are practically one-dimensional, the DP-algorithm performed on average marginally worse than strategy 1b with dual bid-prices, but in the most difficult constraint setting C2, the DP-algorithm performed on average 2.3% better than strategy 1b with dual bid-prices.

CONCLUSION

In this thesis, several strategies for the booking problem in air cargo have been discussed. These included using static bid-prices (strategy 1a), time-dependent bid-prices (strategy 1b), a bucket-based strategy with static bid-prices (strategy 2a) and a bucket-based strategy with time-dependent bid-prices (strategy 2b) and a custom dynamic programming algorithm.
The strategies have been tested in two settings. In the first setting, MIPs were used to ascertain the improvement potential in performance of the various extended strategies compared to the basic strategy 1a. In this setting, the instances tackled were small in order to be able to find close to optimal results in the MIPs. In the second setting, larger instances were tackled, however, this has as result that the MIPs could not be used anymore. In order to find well-working decision variables, heuristics were used.
In this conclusion, the research question as posed in Chapter 2 will be answered, which was the following:

How well does a strategy of using a single EC or bid-price perform in the booking problem compared to the a posteriori optimum, in terms of expected cumulative returns obtained? Is it possible to increase expected cumulative returns by implementing more advanced strategies for the booking problem?

By means of MIPs, it was found that in the most realistic capacity setting, namely pure volume-constrained, using time-dependent EC's based on volume (s1b) improved average cumulative SCb by 3 to 18 percent compared to a strategy using a single bid-price based on volume (s1a), dependent on dataset used and number of requests used. The larger the number of requests, the larger the increase. In a realistic, non-clairvoyant, practical case, the respective increase using solutions obtained by heuristics was 1 to 5 percent.

For a bucket-based strategy (s2a), the increases compared to s1a ranged 2 to 4 percent in the MIP-case and 0.6 to 0.7 percent in the heuristic case.
Combining time-dependent ECs and buckets in s2b, the increases compared to s1a ranged 5 to 11 percent in the MIP-case. This corresponds to a closure of the optimality gap between the a posteriori optimum and strategy s1a of 60 to 65 percent. Heuristics were not attempted for this strategy. It should be noted that for the bucket-based strategies only small instances were considered, thus the maximal increase is smaller than the maximal

increase in the case of s1b where larger instances were also considered. On the small instances, increases by using s1b in the MIP-case ranged 3 to 10 percent.
This study has confirmed that combining buckets and time-dependent EC's leads to theoretically better performing strategies for a knapsack problem with stochastic requests. The increase in expected returns by using this strategy exceeds the individual increases by using the composing strategies, i.e. s1b and s2a. This shows that there is some synergy between the two composing strategies and that these strategies do not tackle exactly the same problem by different means.

However, obtaining decision variables for this strategy, i.e. constant bucket proportions and time-dependent EC's at the same time, that perform robustly well over changing scenarios is extremely difficult, even when knowing the exact distributions of request's characteristics: returns and capacity requirements .Thus, the promising theoretical gains may be extremely difficult to achieve in practice.
As above results show, amongst the two composing strategies, a strategy of time-dependent EC's (s1b) performed better in both theoretical as practical settings than a bucket based strategy with static EC (s2a). Thus, it seems to hold greater potential for performance gains next to being much easier to determine the optimal decision variables for.

In the case of a flip-flop setting, a strategy of dual bid-prices outperformed a bucket-based strategy in a theoretical setting, but was outperformed by strategies using time-dependent bid-prices based on volume only. However, well-performing dual bid-prices are not difficult to obtain in a heuristic manner. In practical cases, these solutions outperformed the heuristic solutions of both time-dependent bid-prices based on volume and a bucket-based strategy based on volume.
Interestingly enough, in this setting the best performing strategy in practical cases was the DP-algorithm described in this thesis. This confirms the potential of this approach in nontrivial multidimensional problems, i.e. problems where none of the dimensions are redundant. Also, the DP-algorithm has negligible running times in a practical approach and there is no determining of optimal decision variables, such as bid-prices, involved. The algorithm only takes as input the current request, which is to be accepted or rejected, and assumed distributions of requests' characteristics.
In one-dimensional cases the DP-algorithm performed marginally worse than a strategy based on time-dependent bid-prices.

It was also found that in both a non-trivial two-dimensional case as in a one-dimensional case a strategy based on dual bid-prices is outperformed by s1b based on volume. Thus, it would seem that the benefit of using dual bid-prices is smaller than the benefit of using time-dependent bid-prices, and this could make the decision approach of determining bid-prices simpler in practice.
However, this result is very dependent on the specific characteristics of air cargo, namely a strong correlation between returns per m^3 and SCb per kg due to a relatively stable density, and might not hold in other applications.

Managerial implications:

It was found that time-dependent EC-setting offered a larger increase in expected returns than implementing buckets, compared to a static EC-setting. Theoretically, we have found that combining time-dependent EC's with buckets offers an even larger increase, but it was deemed too complex to find the combination of time-dependent EC's and bucket proportions for this strategy. It is advised that more research is put into the setting of these variables before this strategy is implemented.

It is not difficult to improve on a static EC-setting by implementing time-dependent EC's. In this paper, a simple method of using median optimal EC's by solving a mathematical program repeatedly led to EC's that outperformed the strategy of using the optimal static EC.

In the case where the booking problem has multiple dimensions, such as in the case of air cargo with weight and volume, and none of the dimensions is redundant in practice, it might be worthwhile to investigate the use of the DP-algorithm proposed in this paper. In multidimensional cases, this DP-algorithm performed even better in terms of expected cumulative SCb than the strategy of using time-dependent EC's. However, this DP-algorithm does not use bid-prices at all and thus it can probably not easily be applied in the current airlines' booking request acceptance policy.

DISCUSSION

Various limitations exist in the research performed in this thesis, the most important of which are the following:

- For generating requests, real-life AWB data was used. However, real-life AWB's mean that these are requests that have been accepted in the past. No data was available on requests that had been rejected. Thus, the dataset is biased vis-à-vis the unknown dataset of real-life requests, in the sense that the dataset used on average has too high returns per capacity units. This can clearly be seen in Figure 7, where the class with SCb/m^3 less than 100 is almost completely absent, because requests having this SCb/m^3 were almost always rejected. However, this downside could be partly tackled by varying the amount of requests, as by having a dataset that is biased positively but taking the number of requests biased negatively, the two might cancel out. This was not possible in this research as the average number of requests per flight was also unknown. However, all used strategies benefit from this bias. If it is assumed all strategies benefit equally from the bias, the relative increases mentioned in this thesis are still exactly valid. This limitation is more a barrier for determining optimal bid-prices and bucket proportions in practice. ECs mentioned in this thesis might not be realistic or optimal in real-life because it was not possible to mimic the real-life setting exactly.
- The number of requests was hold constant and was not stochastic. The number of requests was set at a value consistent with used capacity settings such that the flight would be somewhat constrained.
- The instances analyzed were relatively small, namely 20 requests. However, results for strategy s1b have shown that larger instances only offered larger potentials for return increases in the case of more complex strategies.
- Recourse strategies were not investigated for bid-price models. For instance, if in a scenario the remaining capacities at the end of the booking period were still very large, EC's were kept at positive levels where it probably would be better to set them at zero.

Suggestions for further research include the following:

- Extending strategies s1b, s2a and s2b to also use dual bid-prices.
- Investigating methods to solve large MIPs faster, such as Lagrangian relaxation, which could especially be beneficial in the case of the MIP for strategy s2b.
- This thesis was focused on the theoretical analysis of the increase in expected cumulative SCb when adapting more complex strategies than a simple bid-price strategy. Heuristics have been attempted in practical situations but were not the focus of this thesis. It is likely that better performing heuristics can be found.
- Investigate recourse strategies. Recourse strategies might be able to further close the optimality gap between the analyzed bid-price strategies and the a posteriori knapsack optimum. It might be difficult to include recourse strategies in a MIP, but it is not difficult to include it in the simulation.
- The decision rule in the DP-algorithm used was to accept a request if the expected remaining cumulative SCb when accepting the request was larger than when rejecting the request. It is possible that the DP-algorithm would perform better when using a different decision rule, for instance, accepting a request if in the majority of future scenarios the cumulative SCb is larger in this case than when rejecting the request. It is not likely that this specific adjustment would perform better, but only one decision rule was tested.

Bibliography

Ahmed, S. & Shapiro, A., 2002. *The Sample Average Approximation Method for Stochastic Programs with Integer Recourse,* Atlanta, GA: School of Industrial & Systems Engineering, Georgia Institute of Technology.

Amaruchkul, K., Cooper, W. L. & Gupta, D., 2007. Single-Leg Air-Cargo Revenue Management. *Transportation Science,* 41(4), pp. 457-469.

Frieze, A. & Clarke, M., 1984. Approximation algorithms for the m-dimensional 0,1 knapsack problem: worst-case and probabilistic analysis. *European Journal of Operations Research,* Issue 15, pp. 100-109.

Han, D. L., Tang, L. C. & Huang, H. C., 2010. A Markov model for single-leg air cargo revenue management under a bid-price policy. *European Journal of Operational Research,* Issue 200, pp. 800-811.

Huang, K. & Hsu, W., 2005. REVENUE MANAGEMENT FOR AIR CARGO SPACE WITH SUPPLY UNCERTAINTY. *Proceedings of the Eastern Asia Society for Transportation Studies,* Volume 5, pp. 570-580.

Kasilingam, 1996. Air cargo revenue management: Characteristics and complexities. *European Journal of Operational Research,* Volume 96, pp. 36-44.

Moussawi, L. & Çakanyildirim, M., 2005. *Profit Maximization in Air Cargo Overbooking,* Dallas: School of Management: University of Texas at Dallas.

Pak, K. & Dekker, R., 2004. CARGO REVENUE MANAGEMENT: BID-PRICES FOR A 0-1 MULTI KNAPSACK PROBLEM. *ERIM REPORT SERIES RESEARCH IN MANAGEMENT,* pp. 1-24.

Papastavrou, J. D., Rajagopalan, S. & Kleywegt, A. J., 1996. The Dynamic and Stochastic Knapsack Problem with Deadlines. *Management Science,* 42(12), pp. 1706-1718.

Rinnooy Kan, A., Stougie, L. & Vercellis, C., 1993. A Class of Generalized Greedy Algorithms for the Multi-Knapsack Problem. *Discrete Applied Mathematics,* Issue 42, pp. 279-290.

Slager, B. & Kapteijns, L., 2003. Implementation of cargo revenue management at KLM. *Journal of Revenue and Pricing Management,* 3(1), pp. 80-90.

APPENDIX A: OVERVIEW OF USED NOTATION IN THEORETICAL MODEL (CHAPTER 5)

Notation	*Explanation*
B_{tf}^{j}	Accepted capacity regarding dimension *j* in period *t,* flight *f*, if no booking was accepted in period *i*, flight *f,* B_{tf}^{j} equals 0 for all *j*
C_{tf}^{j}	Cumulative expected amount of dimension *j* in flight *f* that will show up of all bookings booked from time *T* to *t+1*
C_{tf}^{jr}	Normalized C_{tf}^{j} (see page 37)
E_{jf}^{+}	The amount of overbooked capacity of dimension *j* that is actually used
f	Index for flight
$O_f(v, w, p)$	Expected offloading costs for flight *f* when having to offload *v* volume, *w* weight and *p* positions
P	Real amounts of positions on the flight plus any overbooked capacity if applicable
Pr_{tf}	Show-up rate of customer associated with request in period *t*, flight *f*, thus expected proportion of requested volume, weight and positions that will show up, assumed constant over all dimensions
P_R	Real amount of pallet positions on the flight excluding overbooking
p_{tf}	Number of remaining pallet positions in period *t*, flight *f*
p_{tf}^{r}	Normalized p_{tf} (see page 37)
q_{tf}^{j}	Dimension j's capacity requirement of period *t*'s booking request for flight *f*
$R(t, v_t, w_t, p_t)$	Optimal total remaining SCb from time *t* to 0, where 0 is the time of departure, given v_t remaining volume, w_t remaining weight and p_t remaining pallet positions in the flight at time *t*
s_t	SCb of request in period *t*
T_f	Period in which the first booking on flight *f* could be placed. 0 is period of departure
t_f	Index for period, for flight *f*
V_f	Operational volume capacity of flight *f* plus any overbooked capacity if applicable
V_{Rf}	Operational volume capacity on flight *f* excluding overbooking

v_{tf}	Remaining volume in period *t,* flight *f*
v_{tf}^{r}	Normalized v_{tf} (see page 37)
W_f	Real weight capacity of flight *f* plus any overbooked capacity if applicable.
W_{Rf}	Real weight capacity on flight *f* excluding overbooking.
w_{tf}	Remaining weight in period *t*, flight *f*
w_{tf}^{r}	Normalized w_{tf} (see page 37)
x_t	In MIPs: equal to 1 if request in period *t* accepted, 0 otherwise

Appendix B: Definitions and Abbreviations

Air-way bill (AWB): A document storing information related to a piece of cargo (including its weight, volume, SCb and date). [29]

Chargeable weight: the weight used in pricing calculations, as opposed to actual weight. The chargeable weight of indense goods is corrected to make up for their relatively large volume.

Constrained flight: A flight that is (very close to) full on one of the capacities.

Days of week (DoW): weekdays, the different possible days in a week to fly a flight.

Entry condition (EC): Threshold condition for SCb set by Revenue Management. If SCb for a segment is lower than the EC for that segment, the quote is generally rejected.

FCFS: First-come-first-served, equal to an EC-setting with EC of zero.

Flip-flop flight: constraint setting C2, i.e. a flight that is sometimes volume constrained, sometimes weight constrained.

Forwarder: Direct customers of cargo airlines, acting as intermediaries between airlines and end customers.

Interline: A segment of a shipment carried by a different airline than the shipping airline, for a fee paid by the shipping airline.

Itinerary: the exact travel route from point A to B, including transshipment points and modes of transport.

Operational volume: the volume that a good practically takes up, which is potentially different from its water volume (for instance, due to restrictive stacking rules).

Rock-bottom entry condition: The lowest possible level of entry condition, equal to that flight's variable costs plus a safety margin. Usually used for low-demand flights.

Shipment: A combination of a specification of cargo and an itinerary.

Shipment contribution (SCb): an airline's specification of contribution margin for a segment, consisting of the sum of revenues and surcharges, after deducting variable costs.

Additional Shipment Contribution: equal to a flight offer's price minus that flight's entry condition. If this value is positive in principle the offer is accepted and if negative it is rejected. However, this decision rule might not be optimal.

APPENDIX C: PSEUDOCODE FOR ALGORITHMS

Algorithm S_f^1:

This algorithm finds all states to be visited in order to evaluate $R_f(t, v_{tf}, w_{tf}, p_{tf}, C_{tf}^v, C_{tf}^w, C_{tf}^p)$ in a matrix S_f. The algorithm also takes S_f as input and an index *in*. S_f should be initialized to S_f = $[t, v_{tf}, w_{tf}, p_{tf}, C_{tf}^v, C_{tf}^w, C_{tf}^p]$ and *in* to 1.

Algorithm S_f^1
Input: $t, v_{tf}, w_{tf}, p_{tf}, C_{tf}^v, C_{tf}^w, C_{tf}^p$, S_f, in
Output: S_f

if t>0
  if row $[t-1, v_{tf}, w_{tf}, p_{tf}, C_{tf}^v, C_{tf}^w, C_{tf}^p]$ not yet in S_f
    $S_f :=$ S_f $[t-1, v_{tf}, w_{tf}, p_{tf}, C_{tf}^v, C_{tf}^w, C_{tf}^p]$
  endif
  if t > 0 and $v_{tf} - B_{tf}^v \geq 0$ and $w_{tf} - B_{tf}^w \geq 0$ and $p_{tf} - B_{tf}^p \geq 0$ and row
  $[t-1, v_{tf} - B_{tf}^v, w_{tf} - B_{tf}^w, p_{tf} - B_{tf}^p, C_{tf}^v + Pr_{tf}B_{tf}^v, C_{tf}^w + Pr_{tf}B_{tf}^w, C_{tf}^p + Pr_{tf}B_{tf}^p]$
  not yet in S_f
    $S_f :=$ S_f $[t-1, v_{tf} - B_{tf}^v, w_{tf} - B_{tf}^w, p_{tf} - B_{tf}^p, C_{tf}^v + Pr_{tf}B_{tf}^v, C_{tf}^w + P_{rtf}B_{tf}^w, C_{tf}^p + Pr_{tf}B_{tf}^p]$
  endif
endif
if in $\leq$ number of rows of S_f
  $S_f := S_f^1(S_f(in, :), S_f, in+1)$ *** where $S_f(i, :)$ is the i-th row of S_f
endif

Algorithm 1: S_f^1

Note that states for which v_{tf}, w_{tf} and/or p_{tf} are negative are not added. These states always have expected remaining SCb of minus infinity, so they do not need to be kept track of individually.

Algorithm S_f^2:

Algorithm S_f^2

Input: $t, v_{tf}^r, w_{tf}^r, p_{tf}^r, C_{tf}^{vr}, C_{tf}^{wr}, C_{tf}^{pr}$, S_f, V_f, W_f, P_f, in

Output: S_f

If t>0

if row $[t-1, v_{tf}^r, w_{tf}^r, p_{tf}^r, C_{tf}^{vr}, C_{tf}^{wr}, C_{tf}^{pr}]$ not yet in S_f

$S_f :=$ S_f $[t-1, v_{tf}^r, w_{tf}^r, p_{tf}^r, C_{tf}^{vr}, C_{tf}^{wr}, C_{tf}^{pr}]$

endif

if $v_{tf}^r - \frac{B_{tf}^v}{V_f} \geq 0$ and $w_{tf}^r - \frac{B_{tf}^w}{W_f} \geq 0$ and $p_{tf}^r - \frac{B_{tf}^p}{P_f} \geq 0$ and row $\left[t-1, r\left(i, v_{tf}^r - \frac{B_{tf}^v}{V_f}\right), r\left(j, w_{tf}^r - \frac{B_{tf}^w}{W_f}\right), r\left(k, p_{tf}^r - \frac{B_{tf}^p}{P_f}\right), r\left(l, C_{tf}^{vr} + \frac{Pr_t B_t^v}{V_f}\right), r(m, C_{tf}^{wr} + \frac{P_{rt} B_t^w}{W_f}), r(n, C_{tf}^{pr} + \frac{Pr_t B_t^p}{P_f})\right]$ not yet in S_f

$S_f :=$ S_f $\left[t-1, r\left(i, v_{tf}^r - \frac{B_{tf}^v}{V_f}\right), r\left(j, w_{tf}^r - \frac{B_{tf}^w}{W_f}\right), r\left(k, p_{tf}^r - \frac{B_{tf}^p}{P_f}\right), r\left(l, C_{tf}^{vr} + \frac{Pr_t B_t^v}{V_f}\right), r(m, C_{tf}^{wr} + \frac{Pr_t B_t^w}{W_f}), r(n, C_{tf}^{pr} + \frac{Pr_t B_t^p}{P_f})\right]$

endif

endif

if in $\leq$ number of rows of S_f

$S_f := S_f^1(S_f(i, :), S_f, in+1)$ *** where $S_f(i, :)$ is the i-th row of S_f

endif

Algorithm 2: S_f^2

Algorithm R_f^2:

This algorithm augments matrix S_f with an eighth column representing the expected remaining SCb function for the corresponding row's state.

Algorithm R_f^2
Input: S_f
Output: S_f

in = number of rows of S_f
while in > 0
    t = S_f(in, 1)
    if t = 0
        S_f(in, 8) = $-O(S_f(\text{in}, 5)V_f - V_{Rf}, S_f(\text{in}, 6)W_f - W_{Rf}, S_f(\text{in}, 7)P_f - P_{Rf})$
    else
        a = -∞
        if row $\left[t-1, r\left(i, v_{tf}^r - \frac{B_{tf}^v}{V_f}\right), r\left(j, w_{tf}^r - \frac{B_{tf}^w}{W_f}\right), r\left(k, p_{tf}^r - \frac{B_{tf}^p}{P_f}\right), r(l, C_{tf}^{vr} + \frac{Pr_t B_{tf}^v}{V_f}\right), r(m, C_{tf}^{wr} + \frac{Pr_t B_{tf}^w}{W_f}), r(n, C_{tf}^{pr} + \frac{Pr_t B_{tf}^p}{P_f})]$ exists in S_f
            a = s_{tf} + S(corresponding row index, 8)
        endif
        index = row index of $[t-1, v_{tf}^r, w_{tf}^r, p_{tf}^r, C_{tf}^{vr}, C_{tf}^{wr}, C_{tf}^{pr}]$ in S_f
        b = S_f(index, 8)
        S_f(in, 8) = $\max(a, b)$
    endif
    in := in-1
endwhile

Algorithm 3: R_f^2

Algorithm A_f^2:

This algorithm augments matrix S_f with a ninth binary column with a value equaling one if in the revenue-optimizing strategy the offer at time *t* should be accepted, 0 otherwise. Note that for *t*=0, the value is arbitrarily set to zero, because there is no offer at this time.

Algorithm A_f^2
Input: S_f
Output: S_f

$S_f(:,9) = 0$
in = last row index of S_f for which $t \neq 0$
while in > 0
    index = row index of S_f for which $S_f(index, 1) = S_f(in, 1) - 1$ and $S_f(index, q) = S_f(in, q)$ for q = 2,3…, 7.
    if $S_f(index, 8) \neq S_f(in, 8)$
        $S_f(in, 9) = 1$
    endif
    in = in-1
endwhile

Algorithm 4: A_f^2

Algorithm D_f^2:

The following algorithm outputs a T-length binary vector with the *t*-th position equaling 1 if the offer at time *t* should be accepted, 0 otherwise.

Algorithm D_f^2
Input: S_f
Output: D_f
Index = 1
D_f = empty *T*-vector
for t = 1:T
in = row index of row of first seven columns of S_f for which $S_f(in,1) = S_f(index,1) - 1$ and $S_f(in,i) = S_f(index,i)$ for i = 2,3, …., 7.
inn = row index of row of S_f for which the first seven columns equal $\left[t - 1, r\left(i, v_{tf}^r - \frac{B_{tf}^v}{V_f}\right), r\left(j, w_{tf}^r - \frac{B_{tf}^w}{W_f}\right), r\left(k, p_{tf}^r - \frac{B_{tf}^p}{P_f}\right), r(l, C_{tf}^{vr} + \frac{Pr_t B_{tf}^v}{V_f}\right), r(m, C_{tf}^{wr} + \frac{Pr_t B_{tf}^w}{W_f}), r(n, C_{tf}^{pr} + \frac{Pr_t B_{tf}^p}{P_f})]$ (set inn = 0 if the row does not exist)
if $S_f(index,9) = 1$
$D_f(t) = 1$
index = inn
else
$D_f(t) = 0$
index = in
endif
endfor

Algorithm 5: D_f^2