Dynamic Pricing for Vehicle Ferries: using Packing and Simulation to Optimize Revenues

Christopher Bayliss[1], Christine S.M. Currie[1], Julia A. Bennell[2], Antonio Martinez-Sykora[2]
University of Southampton,
[1]Mathematical Sciences, [2]Southampton Business School
Southampton, SO17 1BJ, UK

Abstract

We propose an improved heuristic approach to the vehicle ferry revenue management problem, where the aim is to maximize the revenue obtained from the sale of vehicle tickets by varying the prices charged to different vehicle types, each occupying a different amount of deck space. Customers arrive and purchase tickets according to their vehicle type and their willingness-to-pay, which varies over time. The optimization problem can be solved using dynamic programming but the possible states in the selling season are the set of all feasible vehicle mixes that fit onto the ferry. This makes the problem intractable as the number of vehicle types increases. We propose a state space reduction, which uses a vehicle ferry loading simulator to map each vehicle mix to a remaining-space state. This reduces the state space of the dynamic program. Our approximation approach allows the value function to be approximated rapidly and accurately with a relatively coarse discretization of states. We present simulations of the selling season using this reduced state space to validate the method. The vehicle ferry loading simulator was developed in collaboration with a vehicle ferry company and addresses real-world constraints such as manoeuvrability, elevator access, strategic parking gaps, vehicle height constraints and ease of implementation of the packing solutions.

Keywords: Revenue management; Packing; Transportation; Dynamic pricing.

1 Introduction

Vehicle ferries are used to transport passengers and their vehicles and for many island populations, they can be the sole means of transporting goods, as well as providing a service to commuters and tourists. The global ferry market in 2012 was valued at over \$15 billion, carrying more than 2 billion passengers and 350 million vehicles. Space on the vehicle deck is typically the binding constraint and managing prices to ensure efficient use of the available space is an important problem. In this article we describe an optimal pricing algorithm that takes into account the efficiency of the packing, and practical considerations such as vehicle manoeuvrability when setting prices for different vehicle types, ranging from large freight vehicles to motorbikes. Customer arrivals into the booking system are stochastic and a customer will purchase a ticket with a probability dependent on the price, their vehicle type and the time left until departure. This assumption allows us to use price as a lever of demand to push towards more efficient and profitable vehicle mixes. While this article is focused on vehicle ferries, a similar problem is encountered in other industries, e.g., sale of advertising time on radio or television channels, setting costs for bespoke manufacturing, and other freight transportation applications.

We use dynamic programming to set prices, where the state of the dynamic program gives an indication of the space remaining on the ferry. To obtain an exact solution, the states of the dynamic program should correspond to the mix of vehicles that have already purchased tickets for the ferry and in previous work we describe how to obtain exact solutions using a combination of mixed integer linear programming and dynamic pricing (see

Martinez-Sykora et al. (2017)). As the size of the ferry and the number of vehicle types increase, the state space can become too large to be easily tractable and it becomes necessary to use an approximation to the space remaining, which we estimate from the output of a simulation of the loading process.

We simulate the loading procedure of the ferry to approximate the amount of space a new vehicle will occupy given the current state of the ferry. This is named a *transition function* (Section 4.4). Transition functions are defined for each deck and return the amount of space taken up by a given vehicle type on that deck including both the area of the vehicle and the expected wasted space it generates, for example by blocking off areas of the deck due to staggered parking. The transition functions derived from the simulation are used in the dynamic program. Our results show that this approach reduces the state space sufficiently to allow a dynamic program to solve real ferry instances, which can accommodate hundreds of vehicles, within a few minutes. The packing simulation model takes account of vehicle ferry loading constraints such as manoeuvrability, elevator access for disabled customers, priority boarding and other case-specific constraints when placing vehicles. Clearly, the capability of the packing procedure to produce good quality configurations of vehicles will have the greatest effect on the amount of wasted space. We have developed a sequential packing algorithm, which chooses positions for each vehicle type remaining to be packed. It then selects which vehicle to load next by maximising a weighted sum of a variety of efficiency based attributes (Section 5.4). A simulated annealing algorithm is used to tune the weights of the algorithm to achieve improved packing efficiency for a given vehicle mix.

Ferries can have flexible space in the form of temporary decks that can be raised and lowered as required between crossings. Typically, making use of these temporary decks will increase the space available for smaller vehicles and reduce the space available for high vehicles. Hence, the decision over whether they should be used or not is dependent on the vehicle mix. We incorporate a method for deciding on the most efficient deck configuration dynamically through the selling period and find that in most cases this out-performs fixing the configuration in advance.

This work builds from our previous work (Bayliss et al. (2016)) and represents a significant development in the scope of the problem definition as well as introducing new methodological tools. Bayliss et al. (2016) only considered a fixed ferry configuration with no height restriction on the main deck. The packing of vehicles was via a fixed loading heuristic and the remaining area was found by a simple numerical approximation. In this work we solve a more realistic problem that incorporates height restrictions on different areas of the main deck arising from the ferry's movable mezzanine decks. This increases the dimensionality of both the packing and dynamic pricing problems, which the new algorithm presented here is capable of handling. Moreover, the new algorithm is able to select the optimal ferry configuration and optimize the prices. The approach described here also improves the optimization of packing decisions by introducing a simulated annealing algorithm that improves the packing efficiency and is less sensitive to the amount of space remaining. Furthermore, the area calculations that form the basis of the mapping of the state space are now calculated exactly via a vehicle sliding procedure (see section 5.2). The improved approach also features a novel gradient-based value function interpolation scheme, which enables us to approximate the value function accurately with a coarse discretization. This is vital for tackling the higher dimensional problem that arises when mezzanine decks are considered. Additionally a variable interval size allows the new model to capture near full capacity ferries more accurately, for example allowing us to capture states where motorcycles still fit onto the ferry when cars will not.

The proposed approach provides a complete pricing and packing solution to the vehicle ferry revenue management (RM) problem which maximizes expected revenues by exploiting the willingness-to-pay distributions, ensuring that customers pay a fare for their vehicle that better reflects the capacity that it uses. The loading simulator is required at every stage of the implementation. It is used first when setting prices, to estimate the remaining space after packing a set of vehicles, which defines the state of the dynamic program. During the selling season, the loading simulator is used to monitor the current remaining-space state, which identifies the prices to offer to each vehicle type in each time step. Using the loading simulator also helps guard against overbooking as it ensures that the accepted vehicle mix is a mix that can feasibly be loaded onto the ferry. At the end of the selling season

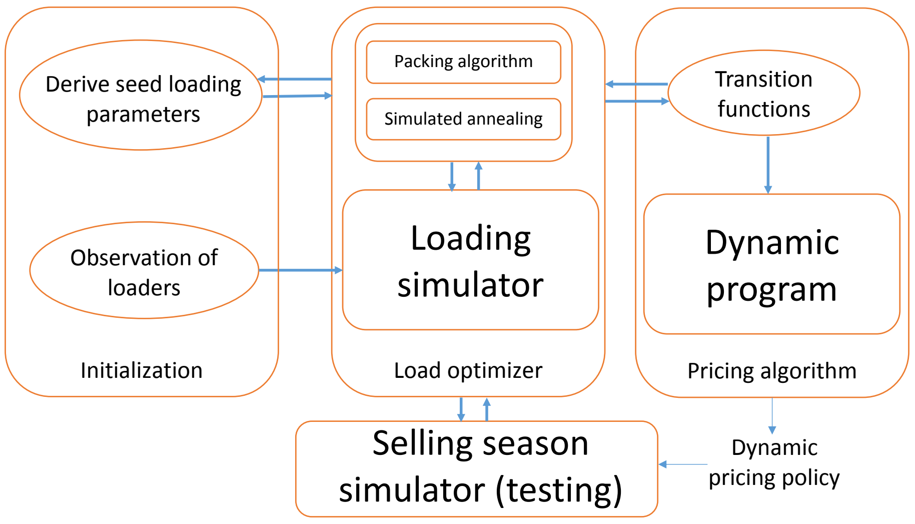

Figure 1: Overview of proposed vehicle pricing optimization framework.

the packing solution can be printed off and given to the loading personnel as a guide.

One of the advantages of the proposed approach is that the computation time is linear in the number of vehicle types and does not suffer from a combinatorial explosion as the number of vehicle types increases.

To summarize, our approach to solving the vehicle ferry RM problem is characterized by five main components, which are also illustrated in Figure 1:

1. Load optimizer: consists of the loading simulator and the packing algorithm. The load optimizer finds the transition functions and calculates the remaining space for any given vehicle mix during the selling season. It can also be used to pack vehicles efficiently.

   (a) Loading simulator: a simulation model of the vehicle loading process. The simulation was developed after observing the loading process first hand and discussing it with personnel from the firm, to ensure that it captures the most important practical aspects of loading. For example, unrestricted access to the elevators, parking gaps and ease of implementation of the packing solution. See Section 5 for details.

   (b) Packing algorithm: rules for vehicle selection and placement used within the simulator. Simulated annealing is used to tune the parameter values. See Section 5.4 for details.

2. Transition functions: describe the space requirements of different vehicle types. These are used as the input for the dynamic program. See Section 4.4 for details.

3. Dynamic program: used to find the optimal dynamic pricing policy (Look-up table policy which specifies a price for each vehicle type in each time period at each remaining space state). See Section 4.1 for details.

4. Selling season simulator (testing): a simulation of the selling season is used to test the policies.

The remainder of the paper is organized as follows. We review the relevant literature in Section 2, before describing the practical problem we are solving in more detail in Section 3. In Section 4 we formulate the pricing problem, in Sections 5 we introduce the load optimizer. Results are presented in Section 6 and we draw some conclusions and discuss future work in Section 7.

2 Related Literature

2.1 Dynamic Pricing

Dynamic pricing regulates sales via automatic price adjustments in response to statistical fluctuations in demand. The earliest work in the area was carried out by Kincaid and Darling (1963) but the past two decades have seen an increase in the number of applications of dynamic pricing and the complexity of the problems being solved. We focus on applications of dynamic pricing where capacity is limited and products are perishable. Previous applications include airline ticket pricing (e.g. Currie and Simpson (2009)); fashion and retail Bitran and Mondschein (1997); car rental Li and Pang (2017); railways Bharill and Rangaraj (2008); delivery prices for online groceries Yang et al. (2016); air cargo RM (Kasilingam, 1996; J.S.Billings, 2003; Amaruchkul et al., 2007) and cruises Maddah et al. (2010). The final two applications have some relevance to the ferry problem as we discuss below. A more comprehensive guide to the applications of dynamic pricing can be found in Talluri and Ryzin (2004).

We have found no other work related to RM in the vehicle ferry industry but (Maddah et al., 2010) apply RM to optimal pricing of cabins on cruise ships. With a cruise ship, there are two competing constraints: lifeboat capacity and cabin capacity. In this sense, the cruise ship RM problem is similar to the vehicle ferry RM problem, although in our example, the lifeboat constraint is rarely binding and the capacity constraint for the vehicles is continuous and two or three-dimensional (dependent upon the need for height constraints) rather than a discrete and one dimensional state definition, which is the case for cabins. Both problems have a large state space, which the authors of (Maddah et al., 2010) solve using heuristics, including single dimension state space reduction schemes and dynamic programming heuristics. The latter are based on decomposing the problem into two separate dynamic programs where value functions are derived separately for each dimension of the problem, and then combined when the solutions are implemented.

A key characteristic of the price optimization problem for vehicle ferries is that each vehicle type uses different amounts of a multi-dimensional capacity. A similar problem is described in (Xiao and Yang, 2010) who cite examples including container shipping, restaurant RM and air cargo. The authors consider a hypothetical example with two fixed-price products and two capacity dimensions and develop revenue-optimizing accept-reject look-up-table policies. The solution method is a backwards recursion, where a Hamilton-Jacobi equation provides the optimality conditions for deriving the optimal control decision for each discrete remaining-capacity state at each time period. We use a dynamic pricing policy rather than an accept-reject strategy and determine this via heuristics, allowing us to control demand for large, real-world problem instances.

Air freight providers face many of the same capacity constraints as vehicle ferry operators, the main difference being that weight tends to be a more important constraint for air freight and there is greater uncertainty in the dimensions of an order. This uncertainty results in model solutions that need to be more simplified than those that we develop for vehicle ferries. More details can be found in (Kasilingam, 1996; J.S.Billings, 2003; Amaruchkul et al., 2007), where it can be seen that the objective of the optimization in air cargo RM is often to minimize overbooking rather than to optimize revenue. Solution methods used include Markov Decision processes (Han et al., 2010); dynamic and stochastic knapsack (Kleywegt and Papastavrou, 1998b); newsvendor models, e.g., (Wong et al., 2009; Zou et al., 2013).

The formulation of the vehicle ferry RM problem shares some of the characteristics of the dynamic and stochastic knapsack problem Kleywegt and Papastavrou (1998a, 2001) in that items arrive following a stochastic process. In the knapsack formulation, the capacity requirements and reward of each item are random variables, which follow fully defined statistical distributions and are realized on arrival into the system. Dynamic and stochastic knapsack algorithms decide whether to accept or reject an item based on its realized reward and capacity requirements and the capacity remaining in the system. This could be a way of approaching the vehicle ferry RM problem but we have chosen to use pricing as a lever for demand instead of making accept/reject decisions. The vehicle ferry problem also has the additional complication of including the 2-dimensional packing problem of fitting vehicles onto the decks.

The effect of price on demand is an important aspect of optimal pricing models and Chapter 3 of (Philips, 2005) provides a useful discussion of the basic economic ideas, describing the most commonly used demand models. We base our price dependence on the logistic model, in which small variations around the market price can lead to large fluctuations in demand. Other factors can impact on willingness-to-pay, with the time until the end of the selling season being the most common. We include the time-dependence explicitly in the price-response curve (Equation 3) and follow previous authors in assuming that willingness-to-pay increases over time (e.g. see (Anjos et al., 2005; Zhao and Zheng, 2000; Gallego and van Ryzin G., 1994)). We assume myopic customers and, as we consider one sailing at a time, ignore reference price effects (Popescu and Wu, 2007). Other authors have considered strategic customers (e.g. (Lazear, 1986; Dasu and Tong, 2009; Yin et al., 2009; Levin et al., 2010; Su, 2007)).

The optimization problem has a similar structure to that described in Yang et al. (2016), where vehicle routing is integrated into a dynamic pricing algorithm for the e-grocer delivery problem. The solution to the vehicle routing problem is not found explicitly in the paper and the authors assume that it is possible to estimate an insertion cost for each accepted delivery. Here, we have no additional costs associated with allowing a vehicle onto the ferry but we do have a second optimization routine, the packing algorithm, which must be solved alongside the dynamic program.

2.2 Bin packing

Packing vehicles onto the ferry efficiently can be viewed as a 2-dimensional bin packing problem with additional problem specific constraints. In Martinez-Sykora et al. (2017) we have previously considered an exact formulation for the packing problem that uses a mixed integer linear program to optimize the packing. In this work, we consider larger ferries and account for more of the practical constraints that ferry companies face when placing vehicles. For that reason, we use a simulation of the loading process, incorporating a packing algorithm that uses simulated annealing (e.g. see (Aarts and Lenstra, 1997)) to set efficient parameter values.

Heuristics have been used previously in the bin packing literature. In (Beisiegel et al., 2006), a simulated annealing algorithm is implemented for a two-dimensional bin packing problem that arises in the steel industry and solutions are encoded as oriented trees. A parameterized approach to an on-line one-dimensional bin-packing problem has also been considered before by (Hopper and Turton, 2016), who use a genetic algorithm to optimize a policy matrix for loading items. Their algorithms are shown to outperform human generated policy matrices. Our approach is similar but the parameters being optimized in this work are weights that multiply attributes of the currently available parking spaces. An alternative form of heuristic is used by (Trivella and Pisinger, 2016), who consider a multi-level local search algorithm for packing, with the goal of minimizing the distance of the centre-of-mass from a target point.

Rectangle packing is of particular relevance here, given the shape of the vehicles being packed. The current state of the art for 2D and 3D rectangle packing is a genetic algorithm described in Gonalves and Resende (2013). A parameterized approach is used for encoding solutions where the chromosome segments represent the position in the packing sequence and box orientation. The authors note that one of the most time-consuming parts of packing algorithms is the maintenance of the list of available placement positions, which they term Empty Maximal Spaces or EMSs. Here, we deal with this in part using a vehicle sliding procedure (Section 5.2) that regenerates the list of available positions after each placement decision.

The packing problem considered in this paper is a constrained one-dimensional and two-dimensional bin packing problem[1]. A vehicle ferry may have some decks which must strictly adhere to the painted lanes (one-

[1]The pricing problem that is considered in this work has two state dimensions, one for each deck, i.e. one for the one-dimensional bin packing problem on the car deck and one for the two-dimensional problem on the main deck. The pricing problem has an additional dimension for the case when mezzanine decks are in operation on the main deck, which is because the main deck is then split into low and high vehicle regions.

dimensional bin packing), and others catering to a mix of vehicle types where parking is more flexible, making two-dimensional packing patterns permissible. The practical problem we consider in the case study is constrained by the wide variety of hard and soft constraints that must be respected in the vehicle ferry loading process and also because some of the decks are optional and impose stricter height restrictions on some sections of the main deck.

For further details about packing problems see (Lodi et al., 2002) for a survey of two-dimensional bin-packing problems, (Csirik and Woeginger, 1998) for a tour of performance results for on-line packing heuristics and (Wäscher et al., 2007) for an improved typology of cutting and packing problems.

2.3 Approximate Dynamic Programming

While we make use of heuristics and simulation, the optimization method we are proposing here is perhaps best classified as approximate dynamic programming (ADP), because we use a transformation to map from a discrete high-dimensional state to a continuous low-dimensional state. The textbooks of (Bertsekas and Tsitsiklis, 1996) and (Powell, 2007) provide a comprehensive treatment of ADP, which is concerned with finding approximate solutions for problems that have an intractably large state space.

Chapter 12 of Judd (1998) provides a thorough treatment of solving continuous state problems using discretization in which they state conditions for discretization schemes that ensure the same solutions can be achieved with the discretized scheme as with the continuous state space. We use discretization of the state space here, approximating the values of intermediate states using numerical methods where necessary.

Simulation provides a useful tool for reducing the state space in stochastic and dynamic problems. Bertsimas and de Boer (2005) use simulation as a method of state-space reduction in a network RM problem, allowing them to solve practical-sized problems with stochastic and dynamic demand. The simulation is used as part of the revenue approximation and a stochastic-gradient algorithm is implemented to obtain the optimal booking limits.

The work of Farias and Saure (2012) introduces an approximate dynamic programming approach for solving dynamic oligopoly models. The main point of interest for us in this article is that it provides another example of how approximate dynamic programming can be used to address the curse of dimensionality of difficult optimization problems. They use a Mathematical programming formulation for solving the Bellman equations and then approximate the value function with a linear combination of basis functions. A state sampling approach is used to reduce the number of constraints in their mathematical programming formulation.

The use of simulation and heuristics also brings us closer to recent work in simheuristics (e.g. see the review by (Juan et al., 2015)). Much of the previous work in this research area has applied simulation and metaheuristics to solving combinatorial optimization problems but the range of applications is growing beyond the original vehicle-routing problems (e.g., renewable energy (Mallor et al., 2015), social networks (Pérez-Rosés and Sebé, 2015)). Our approach is a little different from the standard simheuristics algorithms, which solve the deterministic problem via heuristics and then test the effect of stochasticity via simulation.

3 Case Study

While the methods described in this paper have wide applicability, the development of the methodology draws on a specific case study from the ferry operator, Red Funnel. Red Funnel operate a vehicle ferry service between the mainland (Southampton) and the Isle of Wight, UK. According to the characterization of RM problems given by (Britan and Caldentey, 2003), the vehicle ferry RM problem that we consider here is a multiple product stochastic demand process with flexible, perishable and continuous capacity and price-sensitive customers. We assume that there are no substitution effects and that customers' willingness-to-pay increases over time, although the same method applies for any time dependent willingness-to-pay distribution, such as stationary or decreasing over time for example.

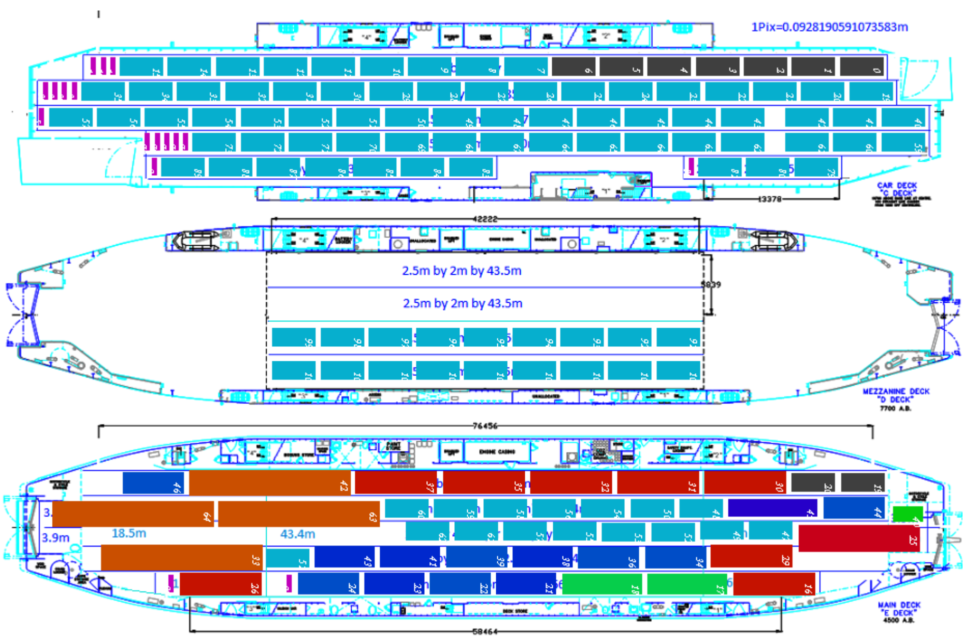

Figure 2: Ferry lane layout diagram with an illustration of packing loss.

Red Funnel have three identical vehicle ferries, each with two permanent vehicle decks, approximately eighty meters long and twelve and a half meters wide. The main deck has a high ceiling (approximately five meters) and accommodates all vehicle types from motorcycles to large freight vehicles and coaches. The upper deck has a relatively low ceiling (approximately two meters) and accommodates cars and motorcycles only. The ferry has two mezzanine decks that can be lowered from the ceiling of the main deck, which accommodate cars only: see the central pane of Figure 2 in which one of the mezzanine decks is in operation. When mezzanine decks are in operation only vehicles below 2.7 metres in height can be parked underneath them on the main deck.

As can be seen in Figure 2, the ferry is nearly symmetrical both laterally and longitudinally, but with the elevators from the vehicle deck only on one side of the ship. These are roll-on-roll-off ferries in which vehicles enter at one end and exit at the other.

We consider thirteen main types of vehicles: cars, motorcycles, vans, minibuses, coaches, medium and large freight vehicles, drop trailers, caravans, other towed vehicles, parcel cages, unaccompanied cars and a miscellaneous category. The price acceptance probability model and the physical dimensions vary between vehicle types.

Vehicle ferry operators face a number of practical constraints on placing vehicles, as listed below.

- Vehicles with hazardous materials must be parked underneath the sprinklers; for Red Funnel these are at the front or the back of the main deck.
- Large vehicles cannot manoeuvre into the corners next to the exits.
- Some customers may require unimpeded access to the lifts.
- Drop trailers (freight vehicles without a towing vehicle) are towed onto the ferry first with a tug and parked in predefined positions.
- Parking gaps are needed to allow vehicles to reverse out of certain positions and passengers to exit the vehicle deck for the duration of the journey.
- Some customers pay an additional charge for priority boarding.

The loading procedure is manually controlled by loaders and yard personnel. Yard personnel sort vehicles into different lanes on the dockside prior to boarding, according to their types and any special requirements, and notify the loaders of the number of vehicles of each type that are to be loaded. Loading of the upper and main decks can take place simultaneously, and cars are loaded onto the upper deck if possible. Loading of the upper deck is straightforward due to the homogeneity of the vehicles, consequently much of the work that follows concentrates on the loading of the main deck.

4 Dynamic Pricing

This section describes the formulation of the pricing problem as a dynamic program. We begin by describing the dynamic program (Section 4.1) before going on to describe the price acceptance model (Section 4.2). The main difficulty in setting up the dynamic program comes in setting up the state space and we focus much of the description below on this issue (Section 4.3). In order to avoid the curse of dimensionality that emerges when states in the selling season are modelled as counts of the number of tickets sold to each vehicle type, we propose mapping these vehicle mix states to remaining space vector states. The loading simulator is used to perform this mapping of a discrete vehicle mix state to a continuous remaining space state by maximizing packing efficiency before calculating the remaining space in each component of the deck space. The approximated continuous states are discretized to facilitate a point-wise approximation of the value function. Transition functions are used within the dynamic program to move from one state to another when a vehicle booking is accepted and we describe how these are calculated in Section 4.4.

4.1 The Dynamic Program

The vehicle ferry RM problem can be formulated as a dynamic program. The arrival rates λ_v, of different vehicle types $v = 1, 2, \ldots, vTypes$, where $vTypes$ denotes the number of vehicle types/vehicle classifications, are derived from historical demand data and are assumed to be constant throughout the selling season. Non-homogeneous arrival patterns could be simulated in a similar way using a time transformation, as in Gallego and van Ryzin G. (1994), but we have not done so here. Customers' willingness-to-pay, or the probability that they will purchase a ticket of price p_v, $\alpha_v(p_v, t)$, is assumed to vary between vehicle types v and with the time remaining until the end of the selling season t as shown in Equation 3. Customers are assumed to be myopic, which is justified for the case where there are few competitors.

Let s' be the new state of the dynamic program, given a purchase is made by vehicle type v when the system is in state s. We discretize time into intervals which are small enough such that it is reasonable to assume that at most one customer will arrive into the booking system during each time interval, i.e. a Bernoulli arrival process Talluri and Ryzin (2004). We solve the dynamic program to find the optimal price to charge $p_v\left(s,t\right)$ for a vehicle of type v, at time t, in state s where the future expected revenue when the system is in state s at time t is equal to:

$$V_t(s) = \left\{ \sum_{v=1}^{vTypes} \lambda_{t,v} \max_{p_v(s,t) \le pMax_v} \left\{ \alpha_v(p_v\left(s,t\right),t)\left(p_v\left(s,t\right) + V_{t-1}(s')\right) + \left(1 - \alpha_v(p_v\left(s,t\right),t)\right) V_{t-1}(s) \right\} \right\} + \left(1 - \lambda_{t,0}\right) V_{t-1}(s). \tag{1}$$

The value of a state is calculated from the values and probabilities of three possible outcomes: 1) a customer arrives and purchases a ticket, the value of which is the sum of the ticket price and the future value of the remaining space state $V_{t-1}(s')$; 2) a customer arrives but does not purchase a ticket; and 3) no customers arrive. The monetary value of the latter is the value of remaining in the same remaining space state for another time period $V_{t-1}(s)$.

The assumption of at most one arrival per time period allows the price optimization to be carried out independently for each vehicle type in each state at each time. Therefore, for each vehicle type v, state s and time t, we need to find the optimal price $p_v^*(s,t)$ for a problem which can be represented as follows

$$\max_{p_v(s,t)\leq pMax_v} \alpha_v(p_v(s,t),t)p_v(s,t)+\alpha_v(p_v(s,t),t)V_{t-1}(s')+(1-\alpha_v(p_v(s,t),t))V_{t-1}(s), \tag{2}$$

As $\alpha_v(p_v(s,t),t)$ is decreasing in $p_v(s,t)$, the first term is unimodal. The following two terms form a linear weighted sum; therefore the pricing sub-problem is unimodal and a Golden Search optimization routine is guaranteed to converge to the optimal solution. Golden search is a numerical optimization technique that iteratively decreases the size of the interval that the optimal solution must lie within. The interval size decreases by the golden ratio in each iteration (See Burley (1974) for a good introduction to Golden Search).

4.2 Price Acceptance Model

We assume the following function for the willingness-to-pay.

$$\alpha_v(p,t) = \begin{cases} \sigma_v\left(\frac{1}{1+e^{\left(k_v\left(\frac{p}{\chi_v}\right)-\psi_v\right)}}\right)\times\left(\gamma_v+(\beta_v-\gamma_v)\left(1-\frac{t}{T}\right)^{\varepsilon_v}\right) & \text{if } p<\chi_v \\ 0 & \text{if } p\geq\chi_v \end{cases} \tag{3}$$

This price acceptance model has two multiplicative components, one for price and another for time. A logistic curve is used to model the price component (e.g., see (Philips, 2005)). The variable χ_v is the maximum price that a customer is willing to pay for that vehicle type; k_v is inversely proportional to the variance of the willingness-to-pay of customers booking vehicles of type v; ψ_v is the mode of the willingness-to-pay distribution relative to the maximum price parameter χ_v; and σ_v is a normalizing factor calculated on the basis that the function returns the maximum probability of price acceptance at the end of the selling season at price zero (assuming willingness-to-pay increases over time). The time component T is the length of the selling period. We base the model on the assumption that the probability of accepting the minimum price is less than or equal to γ_v at the beginning and β_v at the end of the selling season. The time component can capture smooth monotonic effects that time may have on the probability of price acceptance, through the parameter ε_v.

4.3 Discretization of the State Space

We define a mapping from a state with complete information about the vehicle mix X to a reduced state vector $R=\{r_u,r_l,r_h\}$, where R measures the remaining area for vehicles on the upper deck (r_u), the remaining area for low vehicles on the main deck (r_l) and the remaining area for high vehicles on the main deck (r_h) having packed the set of vehicles X. This work uses the following state mapping to reduce the dimensionality of the problem:

$$z(X)\to R \tag{4}$$

The mapping in Equation 4 is performed by the loading simulator introduced in Section 5. This mapping is a fundamental part of our approach for reducing the computational complexity of the dynamic program. State dimension reduction mappings are a common technique in approximate dynamic programming, see Powell (2007).

To facilitate a look-up-table approach for solving the dynamic program in the reduced state space, we discretize each deck space component into a number of discrete levels. The choice of the number of discrete states for each component is a free choice and we show later how accurate estimates of the value function can be obtained using a relatively coarse discretization of the remaining space states. We denote the remaining deck space state dimension components s_i $i\in\{u,h,l\}$. Suppose ζ_i denotes the choice of the number of discrete states for a component r_i of the remaining space state vector R, and μ_i is a parameter that determines how the discretized state interval sizes increase or decrease between the minimum ($rMin_i$) and maximum usable levels ($rMax_i$). Then the relation between the state number s_i for a given component of remaining space r_i is as follows.

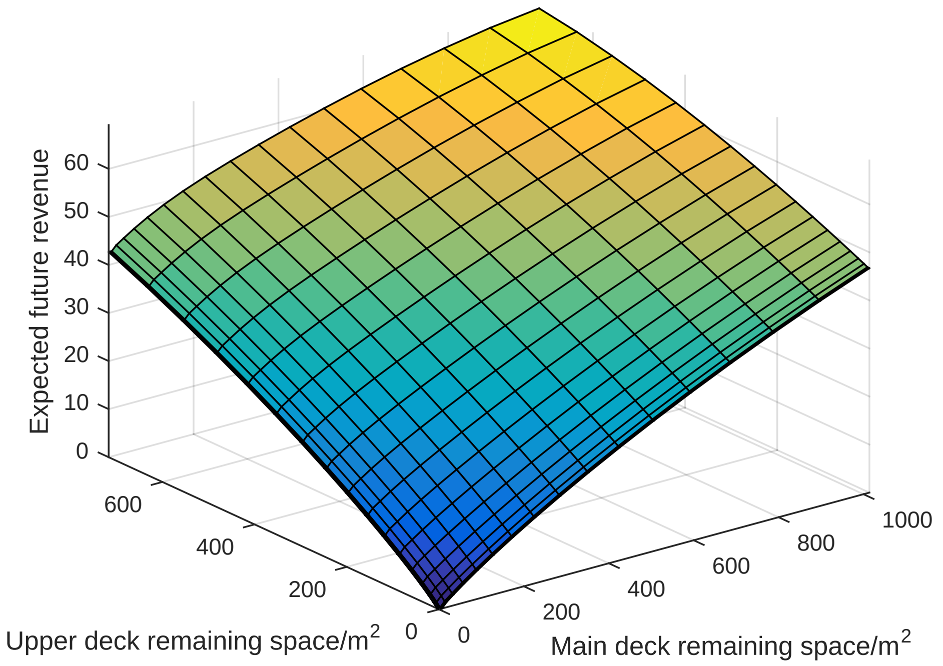

Figure 3: A value function surface at the beginning of the selling season.

$$s_i = \phi(r_i) = \begin{cases} 0 & \text{if } r_i < rMin_i \\ 1 + \pi_i (r_i - rMin_i)^{\mu_i} & \text{otherwise,} \end{cases} \tag{5}$$

Where μ_i is a parameter that can be modified to model increasing or decreasing sizes of remaining space state intervals and π_i is a correction factor equal to $\frac{\zeta_i - 1}{(rMax_i - rMin_i)^{\mu_i}}$ set so that the state number is exactly ζ_i when $r_i = rMax_i$ and the deck space component has not been used at all. The state index s_i is zero for levels of remaining space that are below the minimum usable amount as by definition no more vehicles will fit onto the ferry in such a state. A μ_i value of 1 results in a constant interval size, a μ_i value below 1 leads to an interval size that decreases as the remaining space of deck component i runs out. It is the latter that will be useful in the current context as it allows for the modelling of states where small vehicle types such as motorcycles still fit onto the ferry whilst cars may not. We use $\mu_i = 0.5 \; \forall i \in \{u, h, l\}$ for precisely this reason noting that the approach still works for other values. In the following we use the integer values of s_i between 0 and ζ_i as the states of our dynamic program and omit the deck space component index i for clarity.

Figure 3 illustrates the value function obtained with $\mu = 0.5$, $\zeta_u = 20$ and $\zeta_h = 20$. Where there is no l (low vehicle) remaining space component, this is because the example is one where there are no mezzanine decks in operation, therefore $r_h = rl$ at all times and therefore the number state dimensions is reduced by one compared to the case where there are mezzanine decks in operation. It illustrates the decreasing interval size as space runs out on each deck space component.

To solve the pricing problem we use dynamic programming to compute a pointwise approximation of the value function for each integer state s. When solving the dynamic program for the discrete set of states, Equation 5 is used to find the new state number (s') for state transitions. One of the issues we address in this section is that transitions from discrete states almost always lead to non-integer state numbers. The values of such intermediate states (that are required for solving the dynamic programming equations) have to be interpolated from the values of the neighbouring integer states $\lfloor s' \rfloor$ (floor s') and $\lceil s' \rceil$ (ceil s').

We now describe the numerical interpolation scheme that is used for calculating the values of intermediate states that are encountered when solving the dynamic programming equations. The interpolation method used to

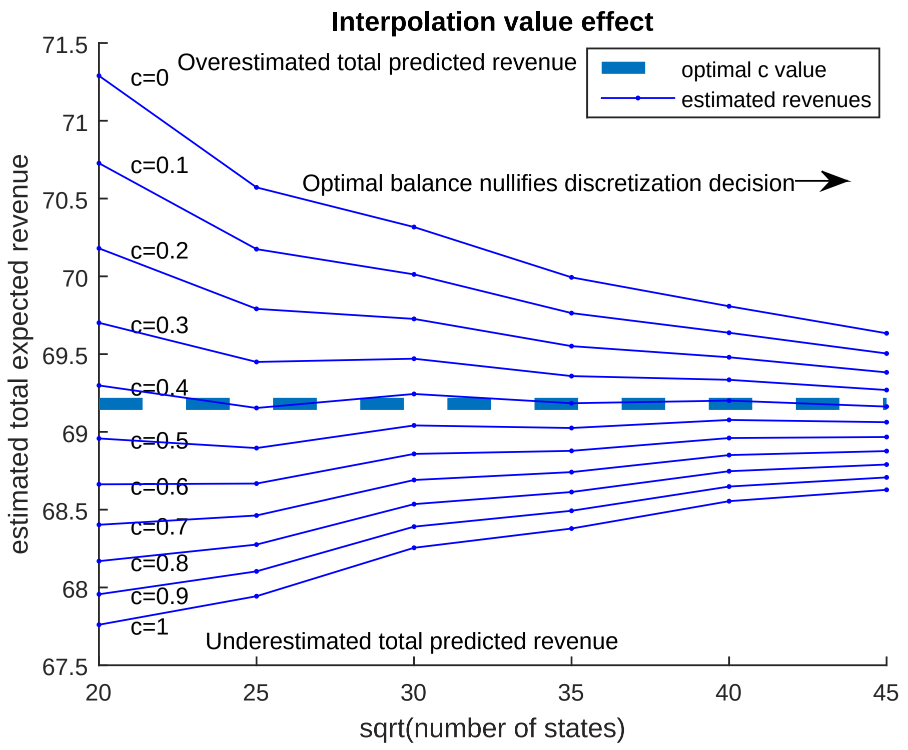

Figure 4: The effect of the interpolation parameter c on the total expected revenue as the number state interval sizes is decreased.

find the value of intermediate states uses a combination of a weighted-sum-of-neighbouring-states approach and a first-order-gradient approach, exploiting the assumption that the value function is concave in the remaining-space state for each time period, to achieve more accurate interpolation than either approach alone. The concavity of the value function (in state index s) is a logical consequence of known results on the asymptotic behaviour of dynamic pricing policies in relation to the available capacity and the duration of the selling season (see Gallego and van Ryzin G. (1994)). The concavity of the value function requires that the following inequality be satisfied $\Delta V_t(\lfloor s \rfloor) \geq \Delta V_t(\lceil s \rceil)$. For our problem this inequality has a logical interpretation: it states that given a fixed amount of time to sell a given amount of capacity, the expected profit for each additional unit of capacity is non-increasing. Furthermore, we demonstrate the concavity with a numerical example, as shown in Figure 3. The first-order-gradient approach estimates the partial derivatives at each defined state of the value function using the values of the neighbouring states. The structure of the value function is such that the weighted sum approach is guaranteed to underestimate the values of intermediate states, whilst the gradient based approach is guaranteed to overestimate the value of intermediate states, and so taking a weighted sum of both estimates results in a more accurate estimate of the values of intermediate states than either approach alone. Note that the benefits of the proposed numerical approach would also apply for a convex value function, or more generally for functions with either non-negative second derivatives or non-positive second derivatives.

Consider an example in which only one dimension of the state space is changing (as is the case when there are no mezzanine decks or a low vehicle can always be placed fully under a mezzanine deck). The interpolated value of the intermediate state s', which is between the discrete interval states $\lfloor s' \rfloor$ and $\lceil s' \rceil$, is given by

$$V_t\left(s'\right) = c\left(d_2 V_t(\lfloor s' \rfloor) + d_1 V_t(\lceil s' \rceil)\right) + (1-c)\left[d_2\left(V_t(\lfloor s' \rfloor) + d_1 \frac{\partial}{\partial s} V_t(\lfloor s' \rfloor)\right) + d_1\left(V_t(\lceil s' \rceil) - d_2 \frac{\partial}{\partial s} V_t(\lceil s' \rceil)\right)\right], \tag{6}$$

where $d_1 = s' - \lfloor s' \rfloor$ and $d_2 = \lceil s' \rceil - s'$ represent the distance of the intermediate state index s' from the neighbouring discrete state intervals $\lfloor s' \rfloor$ and $\lceil s' \rceil$ respectively; c is a constant that weights the contribution of the weighted sum estimate relative to the gradient-based estimate. Our experimentation suggests that for the problem we consider here, if c is set to 0.43, the number of states has a negligible impact on the expected revenue, allowing us to use a coarser discretization without affecting the overall accuracy of the value function approximation. For $c > 0.43$ the estimate of the total expected revenue decreases, for $c < 0.43$ the estimate of the total expected revenue increases. The best value of c in general has to be determined from experimentation for each different problem the approach is applied to; but $c = 0.5$ is a reasonable starting point. Figure 4 demonstrates the effect. The partial derivatives (with respect to state intervals) are estimated numerically at the defined states (integer state indices) using mid-point approximations

$$\frac{\partial}{\partial s} V_t(s) \approx \frac{V_t(s+1) - V_t(s-1)}{2} \tag{7}$$

For the case of state transitions involving the simultaneous use of low and high vehicle space the equivalent of Equation 6 is given in Equation 8. Such transitions occur when mezzanine decks are in operation and a vehicle is parked, on average, in a position that is not entirely under a mezzanine deck and consequently used both high vehicle space and low vehicle space.

$$V_t\left(s_u, s_l', s_h'\right) = c \begin{pmatrix} d_2^l d_2^h V_t\left(s_u, \lfloor s_l' \rfloor, \lfloor s_h' \rfloor\right) \\ + d_1^l d_2^h V_t\left(s_u, \lceil s_l' \rceil, \lfloor s_h' \rfloor\right) \\ + d_2^l d_1^h V_t\left(s_u, \lfloor s_l' \rfloor, \lceil s_h' \rceil\right) \\ + d_1^l d_1^h V_t\left(s_u, \lceil s_l' \rceil, \lceil s_h' \rceil\right) \end{pmatrix} + (1-c) \begin{pmatrix} d_2^l d_2^h \begin{pmatrix} V_t\left(s_u, \lfloor s_l' \rfloor, \lfloor s_h' \rfloor\right) \\ + d_1^l \frac{\partial}{\partial s_l} V_t\left(s_u, \lfloor s_l' \rfloor, \lfloor s_h' \rfloor\right) \\ + d_1^h \frac{\partial}{\partial s_h} V_t\left(s_u, \lfloor s_l' \rfloor, \lfloor s_h' \rfloor\right) \end{pmatrix} \\ + d_1^l d_2^h \begin{pmatrix} V_t\left(s_u, \lceil s_l' \rceil, \lfloor s_h' \rfloor\right) \\ - d_2^l \frac{\partial}{\partial s_l} V_t\left(s_u, \lceil s_l' \rceil, \lfloor s_h' \rfloor\right) \\ + d_1^h \frac{\partial}{\partial s_h} V_t\left(s_u, \lceil s_l' \rceil, \lfloor s_h' \rfloor\right) \end{pmatrix} \\ + d_2^l d_1^h \begin{pmatrix} V_t\left(s_u, \lfloor s_l' \rfloor, \lceil s_h' \rceil\right) \\ + d_1^l \frac{\partial}{\partial s_l} V_t\left(s_u, \lfloor s_l' \rfloor, \lceil s_h' \rceil\right) \\ - d_2^h \frac{\partial}{\partial s_h} V_t\left(s_u, \lfloor s_l' \rfloor, \lceil s_h' \rceil\right) \end{pmatrix} \\ + d_1^l d_1^h \begin{pmatrix} V_t\left(s_u, \lceil s_l' \rceil, \lceil s_h' \rceil\right) \\ - d_2^l \frac{\partial}{\partial s_l} V_t\left(s_u, \lceil s_l' \rceil, \lceil s_h' \rceil\right) \\ - d_2^h \frac{\partial}{\partial s_h} V_t\left(s_u, \lceil s_l' \rceil, \lceil s_h' \rceil\right) \end{pmatrix} \end{pmatrix} \tag{8}$$

4.4 Transition Values

In the dynamic program, state transitions occur every time a customer purchases a vehicle ticket for the ferry. Each vehicle type will use an area in one or more of the upper deck, low space on the main deck, or high space on the main deck. The set of area requirements for each vehicle type in each part of the ferry are referred to as transition values, which will vary between vehicle type v and the area remaining.

We define "packing loss" as the unusable spaces between vehicles. It includes parking gaps, which are required for passengers to exit the vehicle deck, as well as larger gaps that occur when a parking space is blocked off by another (usually larger) vehicle. The transition values include the area of the vehicle and the packing loss. The possibility that some vehicle types can have high transition values in comparison to the area of the vehicle (due to packing loss) allows the pricing algorithm to exploit the opportunities for encouraging demand of easy-to-pack vehicle types, which cause little packing loss, whilst pricing the difficult-to-pack vehicle types according to the space that they consume. As an aside, a quantitative measure of how difficult a vehicle is to pack is the change in the remaining space due to loading a vehicle, divided by the area of that vehicle.

Cars and motorcycles subtract from the remaining space of the upper deck if a sufficient amount of space remains; otherwise vehicles subtract from the space of the main deck. When a high vehicle transition occurs it subtracts by equal amounts from both the remaining low and high vehicle spaces because high vehicle space is also available to low vehicles. When a low vehicle transition occurs the vehicle might not subtract at all from the high vehicle space component. However, when the ferry is busy, it is possible that a low vehicle might be placed in a high vehicle space in some iterations and this may lead to the transition subtracting a small amount from the high vehicle space. Note that if there are no mezzanine decks in operation the low and high vehicle space transitions are always equal and the dimensionality of the pricing problem is reduced by one.

We now define the transition function Equation 9 which maps the remaining space vector R and a vehicle type v to a new remaining space vector R' based on the transition values. We write the transition function as $R' = F(R, v)$, where R' denotes the new remaining space vector, given that the current remaining space vector is R and vehicle type v purchases a ticket. The transition function can be thought of as a method of conveniently packaging the transition values for the purpose of modelling remaining space state transitions.

$$R' = F(R, v) = \begin{cases} r_u \leftarrow r_u - g_u(v) \, \text{if vehicle fits on the upper deck} \\ \begin{array}{l} r_l \leftarrow r_l - g_l(v) \\ r_h \leftarrow r_h - g_h(v) \end{array} \text{otherwise,} \end{cases} \tag{9}$$

where $g_u(v)$, $g_l(v)$ and $g_h(v)$ denote the transition values for a vehicle of type v for space on the upper deck, and for low and high vehicles on the main deck respectively. The transition values were found not to depend on R which is explained by consistent packing efficiency of the load optimizer. The transition values are derived from the loading simulator by recording the space used by vehicles of different types in a set of optimally loaded ferries. In summary the new state s' in (the Bellman) Equation 1 is calculated via the sequence of steps: 1) $R \leftarrow \phi^{-1}(s)$ (Equation 5); 2) $R' \leftarrow F(R, v)$ (Equation 9) and 3) $s' \leftarrow \phi(R')$ (Equation 5).

To simulate different arrival patterns in a selling season, arrivals are generated for each vehicle type v from a Poisson distribution with constant arrival rate, $\lambda_v, v = 1, ..., vTypes$, estimated from historical data on booking numbers. Generating optimally loaded ferries from which to derive transition values involves simulating a selling season, ignoring the impact of prices (which is the same as setting the probability of price acceptance to one), and loading vehicles onto the ferry using our loading algorithm (Section 5.3), where the loading algorithm parameters are re-optimized using simulated annealing when a vehicle fails to fit. If the simulated annealing is unable to find a solution in which the vehicle fits then the vehicle type that arrived last is closed and this process continues until all vehicle types have been closed. No more vehicle arrivals are generated at this point and the loading procedure is repeated with the current set of vehicles, this time measuring transition-values as each vehicle is loaded. We carry out 500 iterations of this procedure starting with a different random seed each time to generate a set of transition values, where 500 was judged to be sufficient because the space requirements of each vehicle remain approximately constant as the number of iterations increases beyond this point. The transition values for each vehicle type are calculated as the average transition values observed in the sample of 500 optimized ferry loads.

5 Load Optimizer

As illustrated in Figure 1 the load optimizer module is central to the proposed approach to pricing vehicles. It is used to estimate the transition function for the dynamic program, to map vehicle mix states to lower dimensional remaining space states and to prevent overselling during the selling season. In each case the role of the load optimizer module is to take as input a vehicle mix and to then pack as many of those vehicles as possible. To achieve this a packing algorithm is used to make placement decisions and a simulated annealing algorithm is used to search the space of the loading parameters for a set of parameters that maximize packing efficiency (Section 5.4). The load optimizer module is used to model ferry capacity during the selling season and therefore packing is re-optimized every time a new vehicle arrives in the selling season. This process is simulated when deriving the transition functions for the dynamic program, and is modelled explicitly when performing selling season simulations to validate the derived dynamic pricing policies.

We categorize the remaining space on the ferry as on-line remaining space or off-line remaining space. On-line remaining space is the area that is reachable from the entrance. Off-line remaining space is the sum of the on-line remaining space for a (possibly hypothetical) minimum-width-minimum-length vehicle plus the unreachable gaps that have been blocked off by other vehicles. On-line remaining space is used in calculating the transition values and for calculating some of the efficiency attributes in Section 5.3, while off-line remaining space is used as a measure of the theoretical maximum remaining space that could be achieved if vehicles were optimally repacked. Off-line remaining space is used for the remaining space states because it allows for the possibility of repacking vehicles more efficiently each time a sale is made.

5.1 Loading Simulator

The loading simulator models the real world constraints of the ferry loading procedure as well as the priority ordering of different vehicle types. For each vehicle mix considered by the load optimizer module the role of the loading simulator is to sequentially choose vehicles to load and where to park them. Algorithm 1 outlines the prioritization of vehicle placement decisions for the loading simulator when it is tasked with loading a given vehicle mix.

Algorithm 1 Loading simulator vehicle prioritization ordering for an input vehicle mix

```
1:  while Vehicles fit and remain do
2:      if Drop trailers remain and predetermined positions remain then
3:          Park these in the predetermined positions
4:      else if Lift access requiring vehicles remain then
5:          Park these next to the lift
6:      else if Cars or motorcycles remain and space remains on the upper deck then
7:          Park these on the upper deck
8:      else
9:          Perform the vehicle sliding procedure for each remaining vehicle type (Section 5.2)
10:         Calculate the highest scoring position for each vehicle type and then based on these calculate the highest
            scoring vehicle (Section 5.3)
11:         Park the recommended vehicle in its recommended position
12:     end if
13: end while
```

The loading simulator is designed to capture the real time sequential nature of the loading process in terms of the reachability of parking positions from the entrance. The physical constraints that must be respected in this process include: minimum parking gaps between vehicles, height restrictions in subregions of a deck due to lowered mezzanine decks, manoeuvrability/large turning circles, which mean that large vehicles should not be parked in either of the corners besides the exit and the fact that vehicles cannot drive through one another. To generate feasible parking positions we use a vehicle sliding procedure (Section 5.2) which traces out an outline of the remaining space (inner-fit polygon) that is reachable from the entrance.

The loading simulator also provides a visual output, as demonstrated in Figure 2. While it is currently being used as part of the pricing algorithm, it is also being developed as a training tool for loading personnel.

5.2 Vehicle Sliding Procedure

This section describes the vehicle sliding procedure that is used to identify the set of reachable parking positions after each vehicle placement decision. The vehicle sliding procedure is a similar idea to an inner-fit polygon, an important method used in irregular shape cutting and packing problems Burke et al. (2007); Bennell and Song (2008). The inner fit polygon is also used to calculate the remaining space on the main deck. This remaining space calculation enables the proposed mapping of a vehicle mix state to a remaining space state. In our sequential loading algorithm the vehicle sliding procedure is run for each vehicle type every time a vehicle is parked to provide candidate parking positions for the next vehicle placement decision.

In the vehicle sliding procedure the sliding-vehicle is initially placed adjacent to and within the ferry entrance and proceeds to slide anti-clockwise around the inside of the ferry and around the outside of vehicles that have already been parked. The sliding-vehicle never travels through gaps narrower than the vehicle itself, or underneath mezzanine decks if the height of the sliding-vehicle exceeds the height of a lowered mezzanine deck, or corner positions besides the exit if the sliding-vehicle has a very large turning circle. The bottom corner positions of the inner-fit polygon are the candidate parking positions for vehicles of the same type as the sliding-vehicle. The pseudocode for the vehicle sliding procedure is given in Algorithm 2.

Algorithm 2 Vehicle sliding procedure for on-line remaining space calculation and finding reachable parking positions

```
1: Place vehicle immediately inside of the entrance let $init\_pos$ denote the initial position and $curr\_pos$ as the current position
2: Let right=0, forwards=1, left=2, backwards=3
3: Let $move(move\_direction)$ be a function that moves a vehicle in the given move direction until it is blocked by another edge or reaches the end of the edge it is currently traversing and then generate an edge on the inner fit polygon
4: Let $at\_corner = true$ if a vehicle reaches the end of the edge it is traversing before it is blocked by another edge
5: $move\_direction = 0$
6: perform $move(move\_direction)$
7: while $curr\_pos \neq init\_pos$ do
8:     if $at\_corner$ then
9:         $move\_direction = modulo(move\_direction - 1, 4)$
10:    else
11:        $move\_direction = modulo(move\_direction + 1, 4)$
12:    end if
13:    perform $move(move\_direction)$
14: end while
15: Generate open positions at the bottom of edges of the inner fit polygon by forwards and backwards move directions
16: Calculate the area of the on-line remaining space by dividing the inner fit polygon into a set of non-overlapping rectangles
```

Algorithm 2 specifies how the vehicle sliding procedure generates a sequence of orthogonal move directions and move distances which trace out the inner-fit polygon. The vehicle sliding procedure requires that the left, right, bottom and top edges of all the parked vehicles and the edges of the ferry are stored in ordered lists based on their relative positions in the dimensions orthogonal to those edges.

Inner-fit polygons are used to calculate the remaining space state. Any polygon consisting of only right angle edges can be divided into a set of non-overlapping rectangles. To do this the extreme edges of the right angle polygon are iteratively cut off until a single rectangle remains. Extreme edges that are targeted are identified as those where the cut leaves a longer edge parallel to the edge just cut off. The remaining space is simply the sum of the areas of all of the cut off rectangles plus that of the final remaining rectangle. Figure 5 illustrates the uses of the inner-fit polygon and the remaining area calculation for a simple example.

5.3 Placing the Next Vehicle

For vehicles that are not parked in predefined positions on the main deck a two phase approach is used to select and place each vehicle. First a single preferred open position is determined for each vehicle type. The preferred position for each vehicle type is selected by scoring the available positions using a weighted sum of a number of efficiency based attribute measurements (Section 5.3.1), and the position with the highest efficiency score is

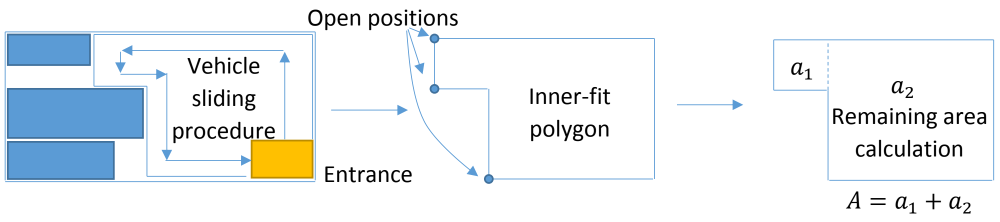

Figure 5: The inner-fit polygon generated from the vehicle sliding procedure is used to identify open positions and to calculate the remaining space for a vehicle.

selected. Second, the decision of which vehicle to load next is made in a similar way. This time a number of efficiency based attribute measurements (Section 5.3.2) are derived from the effect that parking each vehicle type in its preferred position will have on the amount of space that will remain available for the vehicles still waiting to be loaded. The vehicle with the highest weighted sum of attributes is then loaded into its preferred position and the process repeats until all vehicles are loaded or no more vehicles can be loaded. This approach is formalized in Algorithm 3.

The position and vehicle selection attributes are listed and explained below. The list is designed to be exhaustive and several of the attributes are likely to be correlated. For example the position selection attributes *Width inefficiency* and *Tightness* are measurements that both depend upon the width of the given vehicle type but capture different considerations. *Width inefficiency* considers the effect that the width of a vehicle will have on the number of vehicles that can fit next to it after it has been parked, whereas *Tightness* is a measure of how snugly a vehicle fits into a given parking position.

The value for each attribute is normalized.

5.3.1 Position Selection Attributes

1. *Width inefficiency*. Open positions are generated for each vehicle type, and have a maximum width associated with them corresponding to the widest vehicle that can fit within them. Width inefficiency is equal to the total area of vehicles that will fit adjacent to the given vehicle type in the given position (i.e. within the maximum width of the same open position) relative to the total area of vehicles that will then remain to be loaded.

2. *Tightness*. The ratio of the width of the vehicle type plus a standard lateral parking gap divided by the width of the open position. It is interesting to note the contrast between this and the previous attribute.

3. *Entrance overlap ratio*. If a vehicle is parked in the entrance, the entrance overlap ratio is the area of the vehicle type that would overlap the entrance zone, relative to the area of the entrance zone. The entrance zone is defined as the rectangle that begins at the entrance ramp, is as wide as the entrance ramp and is as long as the longest vehicle type.

4. *Uniqueness*. 1 minus the ratio of the number of remaining vehicle types that fit into the open position to the total number of vehicle types that remain to be packed.

5. *Mezzanine deck overlap ratio*. The area of the vehicle type parked directly underneath a lowered Mezzanine deck relative to the area of that vehicle type.

6. *Distance from the bottom*. Longitudinal distance from the exit ramp of the ferry relative to the maximum longitudinal distance of any available position from the exit ramp.

7. *Distance from the right.* Distance from the right hand wall of the ferry relative to the width of the ferry. This is included specifically for the 1 mezzanine deck configuration, where it may be beneficial to park low vehicles under the single mezzanine deck—which is on the right hand side of the ferry—whenever possible, so as to reserve space for high vehicles.

8. *Distance from the nearest side.* Distance from the nearest side relative to the width of the ferry. This measure is included to reflect actual loading practices where some loaders tend to park vehicles close to the sides first before filling in the gaps in the middle.

9. *Adjacent to the side.* 1 if the open position is adjacent to a side wall, 0 otherwise.

10. *Distance from middle.* Distance from the centre of the ferry relative to the width of the ferry.

11. *Width to lowest open position.* Lateral distance to the open position nearest the exit ramp relative to the width of the ferry. This attribute was designed to penalize open positions that block off other open positions that are closer to the bottom (exit).

12. *Bottom adjacency ratio.* In some cases not all of the width of an open position is adjacent to a vehicle parked in front of it, which may lead to staggered parking and the blocking off of other open positions. This attribute is measured relative to the width of the vehicle type the open position is generated for and takes a maximum value of 1. See Figure 6 for more details.

A parking position is selected for each vehicle type based on maximizing a weighted sum of the above attributes. A further set of efficiency attributes are calculated for each vehicle parked in its highest scoring position in preparation for selecting which vehicle to load next.

5.3.2 Vehicle Selection Attributes

1. *Remaining vehicle space.* This attribute provides a single value score for the total floor area of the remaining vehicles that can still fit onto the ferry. It can prevent vehicles from being placed in positions where they obstruct other vehicles from being loaded onto the ferry. Let a'_v denote the "remaining vehicle space" attribute value for vehicle type v and $\xi_{v,v'}$ denote the remaining space for vehicle type v' after parking vehicle type v in its selected position then

$$a'_v = \sum_{j=1}^{vTypes} \xi_{v,j} \left(\frac{n_j area_j}{\sum_{k=1}^{vTypes} n_k area_k} \right),$$

   where n_v is the number of vehicles of type v that remain to be parked and $area_v$ is the area of a vehicle of type v. The calculation of the ξ values is the most computationally expensive part of the loading simulator. As a result, the computational efficiency of the area calculation has a significant impact on the total computation time.

2. *Parking loss.* Measured as the drop in remaining space after parking a vehicle relative to the minimum area required for parking that vehicle.

$$\text{parking loss} = \frac{\text{remaining area before parking} - \text{remaining area after parking} - area_v}{area_v}.$$

3. *Open position selection score.* The weighted sum of the normalised position attributes for the position selected for the given vehicle type, which we reuse for the purpose of selecting which vehicle to park next.

4. *Remaining space for the same vehicle type.* The amount of space remaining for the vehicle type in its preferred position relative to the space remaining for a minimum dimensioned vehicle.

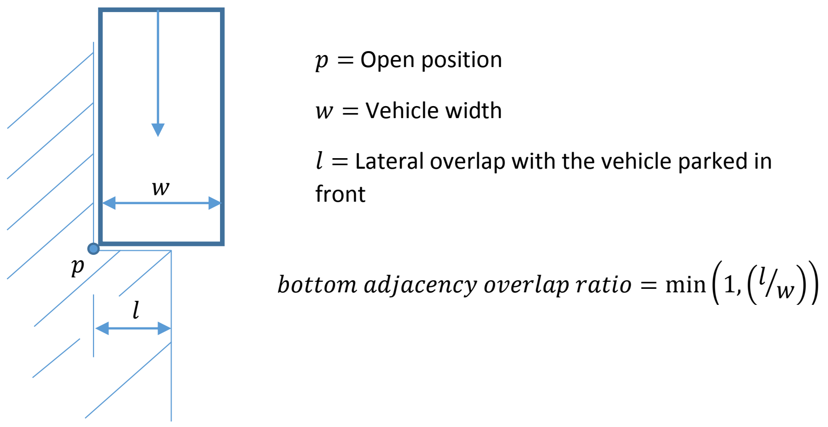
p = Open position
w = Vehicle width
l = Lateral overlap with the vehicle parked in front
w
p
l
$bottom\ adjacency\ overlap\ ratio = \min\left(1, \left({}^{l}/_{w}\right)\right)$

Figure 6: Illustration of the bottom adjacency ratio attribute.

The weighted sum procedure that is used to determine the next vehicle type to load is given in Algorithm 3. The inputs are the sets of available parking positions for each vehicle type and the loading parameter weights. The output is which vehicle to load next (vehicle type l) and where (position index b_l). Note that it is only necessary to consider vehicle types that remain to be loaded in Algorithm 3.

The loading parameters for each attribute (attribute index i) come as a pair of parameters $\{w_{i,1}, w_{i,2}\}$. The single weight for an attribute is calculated as a weighted sum of this pair, where the weight given to $w_{i,1}$ is $1-H$ and the weight given to $w_{i,2}$ is H, where H summarizes the remaining space on the main deck relative to the size of the main deck using the remaining space for low vehicles r_l, which is calculated on line 2 of Algorithm 3. This approach allows the loading algorithm behaviour to change (linearly) over the course of the loading process. Such an approach vastly improves the quality of the packing solutions that can be derived from the packing algorithm.

Algorithm 3 Scheme for selecting the next vehicle to load and in which position

```
1: Calculate the relative remaining space
2: $H = \frac{r_l}{mainDeckSize}$
3: Calculate the position selection scores $\eta_{v,p}$ for each vehicle $v$ in each available parking position $q \in Q_v$ based
   on the position selection attributes $i \in B_q$, where $w_{i,1}$ is the attribute $i$ weight that fully applies when the ferry
   is empty and $w_{i,2}$ is the attribute $i$ weight that fully applies when the ferry is full
4: $\eta_{v,q} = \sum_{i \in B_q} (w_{i,1}(1-H) + w_{i,2}H)\, a_{v,i,q} \forall v \in V, \forall q \in Q_v$
5: Identify the position $b_v$ with the highest score for each vehicle type
6: $b_v = q|\max_{q \in Q_v} \eta_{v,q} \forall v \in V$
7: Calculate the vehicle selection scores $t_v$ for each vehicle in its highest scoring position based on the vehicle
   selection attributes $B_v$
8: $t_v = \sum_{i \in B_v} (w_{i,1}(1-H) + w_{i,2}H)\, a_{v,i,q} \forall v \in V$
9: Select the vehicle $l$ to load next (in its highest scoring position) $l = v|max_{v \in V} t_v$
```

5.4 Optimizing the Parameters of the Loading Algorithm

We use simulated annealing to search for sets of loading parameters that maximize the total area of vehicles that can be loaded and which minimize wasted space. This means that the simulated annealing algorithm influences packing decisions indirectly through the rules that are used to select and place vehicles. Simulated annealing was selected because it is not as evaluation-intensive per iteration as most other metaheuristic algorithms such as genetic algorithms Aarts and Lenstra (1997). It is also a popular method for tackling packing problems (e.g., see Dowsland (1993) who used a method that penalized object overlaps and Martins and Tsuzuki (2008) who also use no-fit polygons to address the packing feasibility issue). Evaluating candidate solutions with the loading simulator is the main computational bottle neck, with evaluations taking between 0.01 and 0.1 seconds, dependent on how many mezzanine decks are used, when implemented as a single-threaded Java application, in a i7-4790, with 3.60GHz and 16GB of RAM.

Simulated annealing is used in two stages of the approach outline in Figure 1: firstly, for deriving initial loading parameter solutions, which are stored and used as initial solutions later on, and secondly for optimizing those initial solutions for individual vehicle mixes, which is done when calculating transition values and during selling season simulation testing. The initial solutions are designed to work well for a wide variety of vehicle mixes as to avoid overfitting when setting the transition values and using the algorithm during the selling season solutions can be specific to the vehicle mix and so the neighbourhood structures used differ, as described in Section 5.4.1.

The objective function for the simulated annealing algorithm is to maximize the on-line remaining space (O), minimize the number of blocked off gaps that are big enough to accommodate at least one vehicle type (Q), and minimize the area of the vehicle(s) that could not be loaded (S). The O term maximizes the on-line remaining space to ensure that vehicles are loaded efficiently even if the set of vehicles easily fit onto the ferry. The logic of the Q term is that reducing the number of wasted gaps will help to create larger usable gaps. The S term ensures that in cases where all of the vehicles do not fit onto the ferry, the area of the vehicles not loaded is minimized. These objectives are non-conflicting and each helps to guide a local search algorithm to tightly packed solutions where all of the vehicles are loaded. The objective function can be written as

$$objVal = \sum_{m=1}^{vMixes} \left(O - (M_1 \times Q) - (M_2 \times S)\right)^2 ,$$

where $M_1 = 1000$ and $M_2 = 10$. Although other values can be used, these were chosen because Q is generally a small number and so M_1 is set larger than M_2 for it to have any effect on the search, whilst S is considered a more important aspect of the objective than O and therefore M_1 is set to be greater than 1. Furthermore, since the separate objective terms are not conflicting the exact choice of M_1 and M_2 is non-critical. The summation allows for the case where the interest is in finding a set of loading parameters that work well in each of a set of vehicle mix scenarios. Such an approach avoids overfitting and leads to robust parameters. The squaring of the objective value contributions from each vehicle mix ensures that the parameters work uniformly well for each vehicle mix rather than being extremely good in some situations but very bad in others. This is useful when deriving the initial solutions. However, for the cases of estimating the transition values and measuring the remaining space state during a simulated selling season, the focus is on one particular vehicle mix. In each case the simulated annealing algorithm that is used to optimize the loading parameters is that given below in Algorithm 4.

For the case of deriving initial seed loading parameters we optimize the loading parameters $w_{i,1}, w_{i,2}$, $i = 1, \ldots, numPars$ using 2000 iterations of the simulation model, using Algorithm 4 and recording the parameters. This process is repeated 10 times and the best set of loading parameters are stored. For the case where the loading parameters are optimized dynamically during a (real or simulated) selling season for a single vehicle mix, we always start with the stored initial loading parameters derived earlier. This speeds up the overall computation time, while allowing us to use loading parameters that are suited to the realized demand.

The temperature/annealing scheme (line 5) is based on a linearly decreasing fraction of the absolute value of the current best objective value. This approach allows the temperature scheme to respond to the progress that has been made during the algorithm. A similar approach is considered by Hatami et al. (2015) as this approach is self-calibrating and thus reduces the parameter-setting burden of simulated annealing. In each iteration a random neighbouring candidate solution is generated and evaluated. If all of the vehicles fit onto the ferry the new solution is returned and the algorithm terminates (line 10). If the new solution has an equal or better objective (line 13) than the current best solution it is accepted as the current solution and the best solution, since it is beneficial to accept different solutions with the same objective value to encourage exploration. Otherwise (line 20) the solution is accepted as the current solution with a probability that depends on the temperature and how much worse the

Algorithm 4 Simulated annealing algorithm for the optimization of the attribute weights

```
1: input: current attribute weights $w$, vehicle mix that current weights cannot load
2: output: new attribute weights
3: Set simulated annealing parameters: $maximum\_iteration$, $c$ (temperature parameter$< 1$), $all\_vehicles\_fit = false$, $current\_objective = best\_objective = evaluate(w_0)$, $iteration = 0$ (iteration count), $n = 0$ (neighbourhood structure), $w = w_0$, $best\_w = w$
4: while $iteration < max\_iterations$ do
5:     $temperature = c\left(1 - \frac{iteration}{maximum\_iterations}\right)|best\_objective|$
6:     neighbour selection: repeat previous improving neighbour type; otherwise generate a new random attribute $i$, random neighbour type $k$, random direction $d$
7:     generate candidate solution $w^{'} = w$ followed by $w_i^{'} = N_{n,i,k,d}$
8:     $objective = evaluate(w^{'})$
9:     if $all\_vehicles\_fit$ then
10:        return $w^{'}$ and terminate
11:    else
12:        $\delta = best\_objective - objective$
13:        if $\delta \leq 0$ then
14:            $w = w^{'}$
15:            $current\_objective = objective$
16:            $best\_objective = objective$
17:            $best\_w = w^{'}$
18:            reset neighbouring solutions $N$
19:        else
20:            if $rand(0,1) < e^{\left(-\frac{\delta}{temperature}\right)}$ or $objective < current\_objective$ then
21:                $w = w^{'}$
22:                $current\_objective = objective$
23:                reset neighbouring solutions $N$
24:            end if
25:        end if
26:    end if
27:    $iteration = iteration + 1$
28:    alternate neighbourhood structure: $n = 1 - n$
29: end while
30: return $best\_w$
```

solution is compared to the current best solution.

5.4.1 Neighbourhood Structure

The simulated annealing algorithm uses two different local neighbourhood structures depending on whether the optimization is being carried out for a set of vehicle scenarios simultaneously (in the case of deriving initial seed solutions) or for the case of optimizing the loading parameters for a single vehicle mix (transition function estimation and state measurements during the selling season). For the former, a step-length based local neighbourhood is used, for the latter *a minimum parameter change* neighbourhood is used. The step-length based approach is useful for aggressively searching the loading parameter space (exploration), whereas the a minimum parameter change neighbourhood is focused on small refinements. For the case of deriving initial seed solutions we consider two varieties of a step-length based local neighbourhood: a linearly decreasing step-length and random step-length. The simulated annealing scheme will choose the linearly decreasing step-length with probability $0.5\left(1 + \left(\frac{iteration}{maxIterations}\right)^{0.5}\right)$, where *iteration* is the current iteration and *maxIterations* is the maximum number of iterations; otherwise the random step-length will be used. This is designed to increase exploration in the early stages and exploitation in the later stages. The linearly decreasing step-lengths are calculated using $SL = 0.5 * \left(1 - \frac{iteration}{maxIterations}\right)$. The step-length can be added or subtracted, for this *sign* is set to -1 or 1 with equal probability.

For the random case *SL* is multiplied by *randInput* to generate a random step-length, where *randInput* is sampled from a Uniform$(0,1)$. A neighbouring solution w' is then generated by randomly selecting a loading parameter index i and an endpoint j to apply the step length.

$$w'_{i,j} = w_{i,j} + sign * SL * randInput.$$

A different neighbourhood is used for the optimization of the loading parameters for a single vehicle mix, which we term *The Minimum Parameter Change Neighbourhood*. This approach is designed for identifying minimal changes to any of the loading parameters which improve loading efficiency whilst retaining desirable features of an incumbent solution. The neighbourhood consists of sets of replacement values for each pair of loading parameters $w^c_{i,1}$ and $w^c_{i,2}$ for each efficiency attribute i, in comparison to the incumbent solution. The replacement values are calculated as the nearest values to the current values that make a difference to at least one decision made during the loading of a particular vehicle mix (Algorithm 3). A change in the value of a loading parameter changes a decision if it causes a different position to be selected for any vehicles or changes which vehicle is loaded next. We consider four alternative minimum change calculations. The first (indexed $k = 0$) considers the nearest parameter value to endpoint 1 of each parameter, $w_{i,1}$, that makes a difference. The second (indexed $k = 1$) considers the nearest parameter value to endpoint 2 of each parameter, $w_{i,2}$, that makes a difference. The third (indexed $k = 2$) considers the minimum common additive change to both endpoints of each parameter w_i that makes a difference. The fourth (indexed $k = 3$) considers the minimum common multiplicative change to both endpoints of each parameter w_i that makes a difference. The notation is defined in Algorithm 3. The equations used for calculating the nearest parameter values are provided in the appendix.

Furthermore, for each case $k = \{0,1,2,3\}$ there is either the absolute minimum parameter changes that are required to change one decision (option $n = 0$), or the average of the minimum parameter changes from the minimum changes required to change the decision that was made to each other possible decision (option $n = 1$). In Algorithm 4, we alternate between $n = 0$ and $n = 1$. The reasoning being that using both diversifies the search: $n = 0$ is more conservative and $n = 1$ is more aggressive. If we also store negative and positive minimum parameter changes separately and index the negative case $d = 0$ and the positive case $d = 1$, we can define the neighbourhood structure $\left(N_{n,i,k,d}\right)$ where i is a specific loading parameter index. This means that a random neighbouring loading solution can be generated (line 6 of Algorithm 4) by generating a random tuple n, i, k, d.

This minimum parameter change neighbourhood allows for the fine tuning of the loading parameter for a particular vehicle mix. In such cases over fitting is not a problem because there is an exact problem that needs a solution.

6 Numerical Experiments

We present results for three different demand scenarios corresponding to: 1. Low proportion of large freight vehicles, high proportion of cars; 2. Average demand rate; 3. High proportion of large freight vehicles, low proportion of cars. Scenarios 1 and 3 correspond to typical sailings in the middle of the day and early in the morning respectively. Each scenario is solved using our method and the pricing approaches described in Section 6.1.

6.1 Alternative Pricing Approaches

We compare our proposed approach with a number of alternative pricing approaches, including an approximation of current pricing practice. For clarity, these alternatives are listed below.

Optimal Fixed Price (*FP*). We find the price that optimizes the revenue equation $\sum_{v=1}^{|V|}\sum_{t=1}^{T} \alpha_{v,p_v,t}\lambda_v p_v$ with the constraint $\sum_{v=1}^{|V|}\sum_{t=1}^{T} \alpha_{v,p_v,t}\lambda_v area_v \leq \Omega$, where $area_v$ is a point estimate of the space required to park a vehicle of type v and Ω is the total space available on the ferry. We find the optimal price to charge using a local search method.

Approximation of current practice (AP). The current approach of the ferry company is constrained by a number of confidential business rules. This prevents us from providing full details of the policy in which a limited number of price changes are permitted during the selling season between a discrete set of price points. Price changes are typically triggered at pre-determined thresholds of remaining capacity, although manual intervention is also used.

To simulate the current approach we implement a dynamic program for the pricing problem identical to that presented in Section 4.1, but with the business rules added as extra constraints. In our implementation, the discrete set of price points for each vehicle type are taken to be equally spaced between the revenue-rate-maximizing or infinite-capacity price (the solution of $\frac{d(pd(p))}{dp} = 0$, where $d(p)$ is the demand as a function of price) and the maximum price χ.

Capacity based pricing (CP). In this approach, each remaining-space state has a set of prices $\mathbf{p}$ associated with it, where the prices are set by observing simulations of the full dynamic pricing policy of Section 4.1. The prices for each remaining-space state are those on offer at the expected time that state is visited. The expected times at which each of the states are visited is calculated by evaluating the full dynamic pricing policy in a forwards time direction starting from a probability of 1 that all capacity is initially available.

Non-fixed ferry configuration (NF). In this approach the decision of the number of mezzanine decks to use is a free variable. The current remaining-space state is measured for each ferry configuration and the current set of prices corresponds to the deck configuration estimated to have the highest expected future revenue. Therefore, the choice of ferry configuration is sensitive to the realized demand during the selling season. This approach can be used in conjunction with any of the pricing policies that are derived from the dynamic program.

6.2 Results

In the scenarios we consider here, we assume that a customer's maximum willingness-to-pay for a particular vehicle type is proportional to its footprint area, which we measure in Car Equivalent Units (CEUs). This assumption does not change the characteristics of the resultant pricing policy but helps to highlight the effect of packing considerations on pricing. Each demand scenario and pricing policy is solved for 0, 1 and 2 mezzanine decks and results of 500 simulations are presented in Table 1. This section focusses on the average revenues achieved by each pricing approach in repeat simulations where the packing problem is solved on-line during the simulations of the selling season. For the interested reader Figure 1 of Appendix Section B illustrates the strong correlation between the expected revenues from the DP look up tables (start of the selling season with all of the space remaining) and those achieved by the pricing policies derived from those value functions tested in 500 repeat simulations.

The computation times for our proposed method with fixed deck configurations range from 12 seconds for the 0 mezzanine deck case with high freight demand to 230 seconds for the 1 Mezzanine deck case with high car demand. The most computationally intensive part of the proposed approach is the derivation of the transition functions, with optimal ferry loading requiring up to 90 seconds for the 1 mezzanine deck configuration and average demand scenario.

Demand	1			2			3		
Ferry configuration	1	2	**3**	1	**2**	3	**1**	2	3
Pricing method	Simulation average revenues 500 repeats								
DP	68.80	70.95	71.12	68.36	**69.24**	66.10	**66.56**	60.77	46.73
CP	67.89	70.38	71.38	67.91	68.56	65.70	66.04	61.28	45.89
AP	65.33	69.23	**71.45**	65.14	67.66	62.00	61.47	54.91	41.18
FP	64.62	66.76	68.76	66.24	66.04	62.69	64.47	51.10	14.36

Table 1: Average revenues achieved by the variants of dynamic pricing.

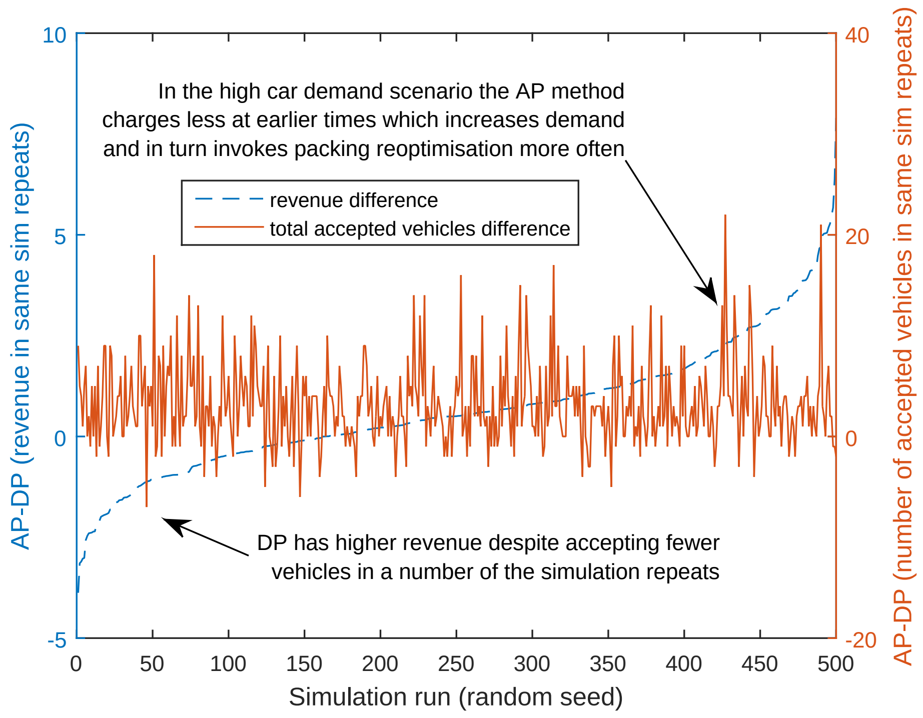

Figure 7: Illustration of the higher ferry utilization of AP in the 2 mezzanine deck high car demand scenario.

Table 1 shows that with two exceptions (D1C3 and D3C2) that our proposed DP pricing approach attains the highest average revenues. It also shows that the 2 mezzanine deck configuration is the most profitable for the high car demand scenario; the 1 mezzanine deck is the most profitable configuration for the average demand scenario; and the 0 mezzanine decks is the most profitable configuration for the high freight demand scenario. These results make sense as these configurations maximize the space available for the vehicle types with the highest expected demand.

Table 1 also shows that with a few exceptions the average revenues for each method of pricing are ordered from highest to lowest average revenue as DP, CP, AP and FP, as introduced in Section 6.1. The notable exception to this is the result that the AP method attains the highest average revenue for the high car demand scenario in the most profitable configuration (2 mezzanine decks). We believe this to be due to more efficient packing rather than improved pricing because this policy reaches capacity more quickly than other pricing policies. Reaching capacity triggers a re-optimization of the packing, improving its efficiency; therefore packing is likely to be more efficient with the AP policy (although computation times will be longer). Using this observation, it may be possible to improve the revenues of the other pricing policies by re-optimizing regularly rather than waiting until one of the vehicle types no longer fits onto the ferry. Figure 7 shows how the AP has a higher ferry utilization than the DP method on average. The FP method performs very badly in the high freight two-mezzanine deck case because is accepts too many freight vehicles which have a habit of blocking the entrance prematurely.

It is worth noting that the 2 mezzanine deck configuration in the high car demand scenario is the simplest pricing problem as the capacity for low vehicles is very high and so a generally good policy is to stimulate a high demand for low vehicles. In the other cases the ideal mix of low and high vehicles is more balanced.

Table 2 shows the effect of the NF (non-fixed ferry configuration) pricing strategy. The NF pricing strategy improves average revenues for the extremal demand scenarios 1 and 3, but leads to reduced revenues for the average demand scenario. Statistical fluctuations in demand in the average demand scenario are such that it is possible that either a 0 mezzanine deck configuration or a 2 mezzanine deck configuration can be the most profitable and therefore form the basis on which pricing decisions are made. However, once a configuration

Demand scenario	Ferry configuration			
	0 M decks	1 M decks	2 M decks	Non-fixed configuration
0	68.80 (0.98)	70.95 (1.00)	71.12 (1.00)	**71.61** (0.99)
1	68.36 (0.99)	**69.24** (0.96)	66.10 (0.99)	67.73 (0.88)
2	66.56 (0.98)	60.77 (0.98)	46.73 (0.92)	**66.58** (0.98)
Average Solution Time (s)	16.05	189.87	129.66	335.58

Table 2: Average revenue from 500 simulation iterations with the optimal revenue policies for each demand scenario written in bold. In brackets are the relative revenues found without re-optimization of the packing.

transition occurs, the new configuration will become an attractor as it will encourage a demand pattern for which the configuration is best suited. The downside of changing configurations it that the final mix of vehicles may not suit the final ferry configuration as well as it could have if the ferry configuration decision was fixed at the beginning of the selling season.

Table 2 shows that for the high car demand scenario and the high freight demand scenario the NF pricing strategy leads to small average revenue improvements. For the high car demand scenario the configuration with the highest expected future revenue appears to systematically vary between the 1 and 2 mezzanine deck configurations. At around time step 600 the probability that the 1 and 2 mezzanine decks have the highest future expected revenue momentarily swap in favour of the 1 mezzanine deck configuration. This observation can be explained as corresponding to the critical point whereby the natural demand for low vehicle types is low or high. If the car demand for low vehicles is below average it will remain profitable to accept bookings from high vehicle types which will typically only be possible in the 1 mezzanine deck configuration. Alternatively if the low vehicle demand is high enough then the 2 mezzanine deck configuration will be the most profitable as this configuration maximizes the space for low vehicle types. In the high freight demand scenario it is rarely the case that the 1 mezzanine deck configuration has the highest expected future revenue, a fact that is reflected in the minor increase in average revenue due to the NF pricing strategy.

Table 2 also shows in brackets the relative average revenues that were achieved when the simulated annealing algorithm is not used to re-optimize the loading algorithm parameters every time a vehicle type is found not to fit onto the ferry. The impact of the simulated annealing re-optimization is an average 2.2% increase in total revenue.

6.3 Comparison to Exact Optimal Solutions

We show that the proposed approach outputs pricing policies that are close to optimal by comparison with results of previous work Martinez-Sykora et al. (2017), where we developed a method that solves the optimization exactly for small problem instances.

In this section we show that the proposed approach provides pricing policies which are close to optimal by comparing the results of the previous work Martinez-Sykora et al. (2017), where we developed a method that solves the optimization problem exactly for problem instances with up to 5 vehicle types for a moderate-sized ferry. To enable the comparison we replace the loading simulator with a 1-d bin packing formulation which simulates optimal lane parking (no 2-d/staggered parking arrangements) as this mimics the packing method used in the exact approach.

One of the approaches for addressing the tractability of the optimal formulation is to discretize vehicle types into a smaller number of categories, so a 5 vehicle type problem can serve as an approximation to a 20 vehicle problem. The discretization requires that the vehicles within each category are modelled as having the maximal dimensions of all vehicles with that category. The following graph shows that for a range of discretizations on the 5 vehicle type problem the proposed simulation based approach stays close to the optimal solution in terms of the average revenue achieved in simulation testing of the pricing policies derived from each approach. Furthermore, Figure 8 shows that allowing 2-d/staggered parking significantly increases revenues. It should be noted that if the

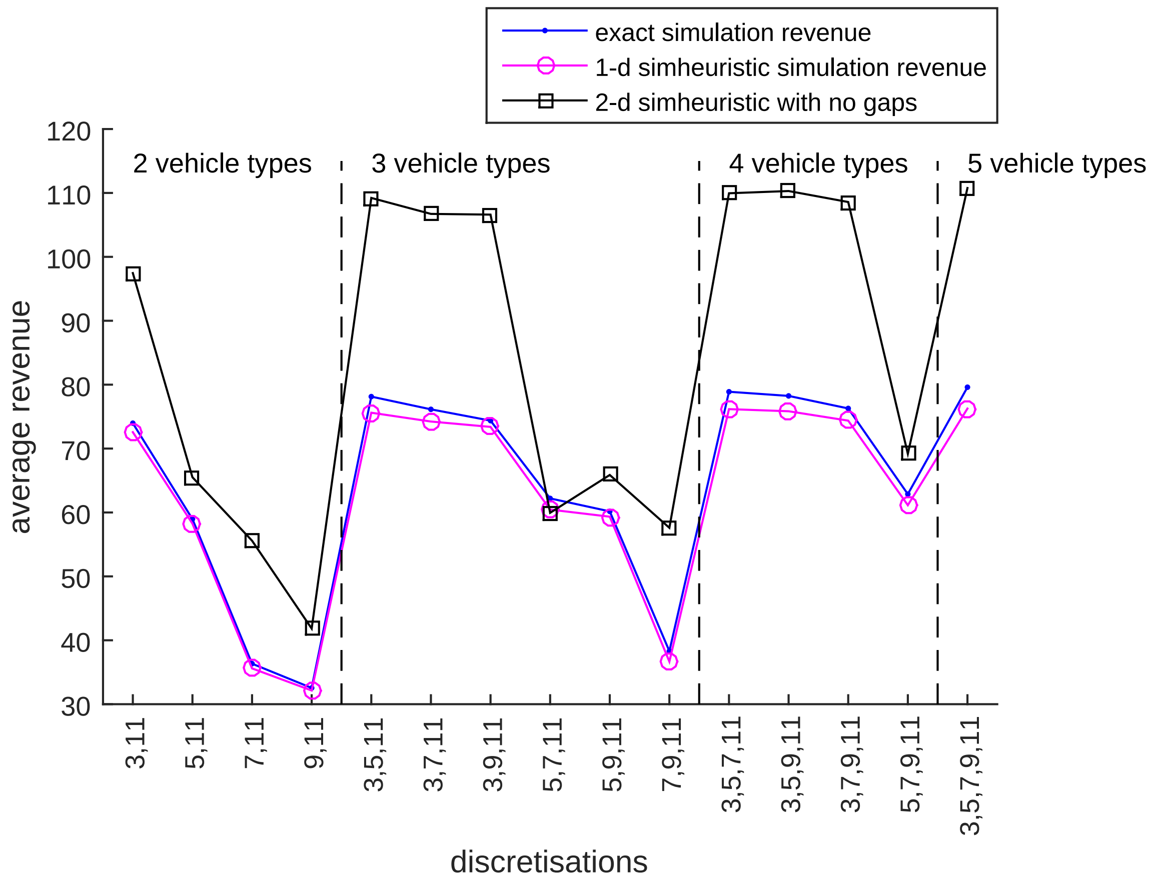

Figure 8: Total average revenue comparison between the proposed simulation-based approach and an optimal formulation for a small problem instance.

optimal formulation utilized a 2-d packing solution the revenue would again be higher than those achieved by the simulation based approach; however the simulation based approach is the only one that remains tractable and close to optimal for large problem instances. The proposed approach took a total of 10 minutes for each discretization, whereas the optimal formulation took over a day to solve. For the 1-d bin packing case the proposed approach achieved on average 97.48% of the revenue of the optimal formulation.

7 Discussion

We have described a practical method for finding the optimal price to charge on vehicle ferries with stochastic demand and variable configurations. This compares well with the exact solution on small problem instances (97.48% of the optimal revenue and 0.7% of the computation time). Our proposed approach also remains tractable for larger problems.

The aim of incorporating the loading simulator was to estimate the remaining space on the vehicle decks of the ferry, and to deduce how the remaining-space decreases as an extra vehicle is added. This helps to approximate the state-space of the dynamic program and allows us to incorporate practical considerations such as manoeuvrability and elevator access. Other structures and optimization routines could be used for the loading algorithm, which may improve computation times and/or its results. This is the subject of future work.

Considerable effort was spent improving the computational efficiency of the simulation model as this is run during the implementation of pricing policies to determine the remaining-space. For reasons of space, the full details of optimizing the performance of the code will not be reported here but the model will be made available online. The simulator has the potential to be useful as a training tool for loading staff or, combined with the loading algorithm, as a way of automating the packing process.

Experimental results from a range of demand scenarios suggest that our proposed approach led to higher average revenues than alternative pricing strategies in most cases. In particular, it attained average revenues that were 6% higher than those attained by the approximation of current practice. We believe this to be due to the method incorporating more flexibility in its pricing, allowing it to react to realized demand and improve the efficiency of the packing. This is underlined by the observation that improving the efficiency of the packing by re-optimizing during the selling season was shown to result in a 2.2% average increase in revenue.

Being flexible with the ferry configuration and allowing it to vary during the selling period was shown to increase average revenue in some cases whilst being susceptible to pitfalls in certain situations. In particular the average demand scenario appeared to have the 0 and 2 mezzanine deck configurations as non-profitable attractors due to the effect that different pricing policies have on the final vehicle mix. Results remain inconclusive about whether the increased computation time associated with this algorithm is warranted given the small increases in revenue that it produces. This may be a result of the demand scenarios that we considered.

The approach we describe here has the potential to be useful in other application areas, e.g. the sale of advertisement slots on radio and television, air freight and bespoke 3-d printing. These share the characteristics that the capacity is continuous and different item types use up different amounts of the space, resulting in a combined pricing and packing problem. In addition, the fact that the case study example has a wide variety of complex features means that it is a simple matter to adapt the proposed approach for other ferry designs which have only a subset of the features which are present in the case study example.

ACKNOWLEDGEMENTS

This work was funded by the EPSRC under grant number EP/N006461/1. We are also grateful to Alexander Strong, Lee Hudson and Pete Smith at Red Funnel, UK who provided us with useful practical advice when designing this method.

REFERENCES

Aarts, E. and Lenstra, J. K. (1997). *Local Search and Combinatorial Optimization.* John Wiley and Sons, Chichester, New York, Weinheim, Brisbane, Singapore and Toronto.

Amaruchkul, K., Cooper, W., and Gupta, D. (2007). Single-leg air cargo revenue management. *Transportation Science*, 41(4):457–469.

Anjos, M. F., Cheng, R. C., and Currie, C. S. (2005). Optimal pricing policies for perishable products. *European Journal of Operational Research*, 166:246–254.

Bayliss, C., Bennell, J. M., Currie, C. S., Martinez-Sykora, A., and So, M.-C. (2016). A simheuristic approach to the vehicle ferry revenue management problem. In Roeder, T. M. K., Frazier, P. I., Szechtman, R., Zhou, E., Huschka, T., and Chick, S. E., editors, *Proceedings of the 2016 Winter Simulation Conference*, pages 2335–2346.

Beisiegel, B., Kallrath, J., Kochetov, Y., and Rudnev, A. (2006). Simulated annealing based algorithm for the 2D bin packing problem with impurities. In Haasis, H.-D., Kopfer, H., and Schnberger, J., editors, *Operations Research Proceedings 2005*, pages 309–314, Heidelberg, Germany. Springer.

Bennell, J. A. and Song, X. (2008). A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums. *Computers and Operations Research*, 1:267–281.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Nashua, USA.

Bertsimas, D. and de Boer, S. (2005). Simulation-based booking limits for airline revenue management. *Operations Research*, 53:90–106.

Bharill, R. and Rangaraj, N. (2008). Revenue management in railway operations: A study of the Radhani Express, Indian railways. *Transportation Research Part A*, 42:1195–1207.

Bitran, G. R. and Mondschein, S. V. (1997). Periodic pricing of seasonal products in retailing. *Management Science*, 43(1):64–79.

Britan, G. R. and Caldentey, R. (2003). An overview of pricing models for revenue management. *Manufacturing Service Oper. Management*, 5(3):203–229.

Burke, E., Hellier, R., Kendall, G., and Whitwell, G. (2007). Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, 179:27–49.

Burley, D. (1974). *Studies in Optimization*. International Textbook Co. Ltd., Leighton Buzzard, UK.

Csirik, J. and Woeginger, G. (1998). *On-line Packing and Covering Problems*, volume 1442 of *LNCS*, chapter 7. Springer-Verlag, Heidelberg, Germany.

Currie, C. S. and Simpson, D. (2009). Optimal pricing ladders for the sale of airline tickets. *Journal of Revenue and Pricing Management*, 8:96–106.

Dasu, S. and Tong, C. (2009). Dynamic pricing when consumers are strategic: Analysis of posted and contingent pricing schemes. *European Journal of Operational Research*, 204:662–671.

Dowsland, K. A. (1993). Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, 68:389–399.

Farias, V. and Saure, D. (2012). An approximate dynmaic programming approach to solving dynmaic oligopoly models. *RAND Journal of Economics*, 43(2):253–282.

Gallego, G. and van Ryzin G. (1994). Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, 40:999–1020.

Gonalves, J. F. and Resende, M. G. (2013). A biased random key genetic algorithm for 2D and 3D bin packing problems. *Int. J. Production Economics*, 145:500–510.

Han, D., Tang, L., and Huang, H. (2010). A markov model for single-leg cargo revenue management under a bid-price policy. *European Journal of Operational Research*, 200:800–810.

Hatami, S., Ruiz, R., and Andreś-Romano, C. (2015). Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. *Int J. Production Economics*, 169:76–88.

Hopper, E. and Turton, B. (2016). Champ: Creating heuristics via many parameters for online bin packing. *Expert Systems With Applications*, 63:208–211.

J.S.Billings, A.G. Diener, B. Y. (2003). Cargo revenue optimization. *Journal of Revenue and Pricing Management*, 2:69–79.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., and Goncalo (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72.

Judd, K. L. (1998). *Numerical methods in economics*. The MIT Press, Cambridge, Massachusetts and London, England.

Kasilingam, R. (1996). Air cargo revenue management: Characteristics and complexities. *European Journal of Operational Research*, 96:36–44.

Kincaid, W. and Darling, D. (1963). An inventory pricing problem. *Journal of Mathematical Analysis and Applications*, 7:183–208.

Kleywegt, A. and Papastavrou, J. (1998a). The dynamic and stochastic knapsack problem. *Operations Research*, 46(1):17–35.

Kleywegt, A. and Papastavrou, J. (1998b). Successive approximations for finite horizon, semi-Markov decision processes with application to asset liquidation. *Operations Research*, 46:17–35.

Kleywegt, A. and Papastavrou, J. (2001). The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41.

Lazear, E. P. (1986). Retail pricing and clearance sales. *American Economic Association*, 76(1):14–32.

Levin, Y., McGill, J., and Nediak, M. (2010). Optimal dynamic pricing of perishable items by a monopolist facing strategic consumers. *Production and Operations Management*, 19:40–60.

Li, D. and Pang, Z. (2017). Dynamic booking control for car rental revenue management: A decomposition approach. *European Journal of Operational Research*, 256:850–867.

Lodi, A., Martello, S., and Monaci, M. (2002). Two dimensional bin packing problems: a survey. *European Journal of Operational Research*, 141:241–252.

Maddah, B., Moussawi-Haidar, L., El-Taha, M., and H.Rida (2010). Dyanmic cruise ship revenue management. *European Journal of Operational Research*, 207:445–455.

Mallor, F., Azcárate, C., and Mateo, P. (2015). Operational management of renewable energy systems with storage using an optimisation-based simulation methodology. *Journal of Simulation*, 9:263–278.

Martinez-Sykora, A., Currie, C. S., So, M.-C., Bayliss, C., and Bennell, J. M. (2017). Optimizing pricing and packing of variable sized cargo. *Under review at Transportation Research Part B*.

Martins, T. C. and Tsuzuki, M. S. G. (2008). Rotational placement of irregular polygons over containers with fixed dimensions using simulated annealing and no-fit polygons. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 30:205–212.

Pérez-Rosés, H. and Sebé, F. (2015). Synthetic generation of social network data with endorsements. *Journal of Simulation*, 9:279–286.

Philips, R. L. (2005). *Pricing and Revenue Optimization*. Stanford University Press, Stanford, Californmia.

Popescu, I. and Wu, Y. (2007). Dynamic pricing with reference effects. *Operations Research*, 55(3):413–429.

Powell, W. B. (2007). *Approximate Dynamic Programming*. John Wiley and Sons, Hoboken, New Jersey and Canada.

Su, X. (2007). Intertemporal pricing with strategic behavior. *Management Science*, 53:726–741.

Talluri, K. and Ryzin, G. V. (2004). *The Theory and Practice of Revenue Management*. Springer Science, New York, USA.

Trivella, A. and Pisinger, D. (2016). The load-balanced multi-dimensional bin-packing problem. *Computers and Operations Research*, 74:152–164.

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research, 183, 109-1130.*

Wong, W., Zhang, A., Hui, Y., and Leung, L. (2009). Optimal baggage-limit policy: Airline passenger and cargo allocation. *Transportation Science*, 43(3):355–369.

Xiao, B. and Yang, W. (2010). A revenue management model for products with two capacity dimensions. *European Journal of Operational Research*, 205:412–421.

Yang, X., Strauss, A. K., Currie, C. S. M., and Eglese, R. (2016). Choice-based demand management and vehicle routing in e-fulfillment. *Transportation Science*, 50:473–488.

Yin, R., Aviv, Y., Pazgal, A., and Tang, C. S. (2009). Optimal markdown pricing: Implications of inventory display formats in the presence of strategic customers. *Management Science*, 55:1391–1408.

Zhao, W. and Zheng, Y.-S. (2000). Optimal dynamic pricing for perishable assets with nonhomogenous demand. *Management Science*, 46(3):375–288.

Zou, L., Yu, C., and Dresner, M. (2013). The application of inventory transshipment modeling to air cargo revenue management. *Transportation Research Part E*, 57:27–44.