Chapter 20
A Dynamic Programming Approach for Integrating Dynamic Pricing and Capacity Decisions in a Rental Context

Beatriz B. Oliveira, Maria Antónia Carravilla and José Fernando Oliveira

Abstract Car rental companies have the ability and potential to integrate their dynamic pricing decisions with their capacity decisions. Pricing has a significant impact on demand, while capacity, which translates fleet size, acquisition planning and fleet deployment throughout the network, can be used to meet this price-sensitive demand. Dynamic programming has been often used to tackle dynamic pricing problems and also to deal with similar integrated problems, yet with some significant differences as far as the inventory depletion and replenishment are considered. The goal of this work is to understand what makes the car rental problem different and hinders the application of more common methods. To do so, a discrete dynamic programming framework is proposed, with two different approaches to calculate the optimal-value function: one based on a Mixed Integer Non Linear Program (MINLP) and one based on a Constraint Programming (CP) model. These two approaches are analyzed and relevant insights are derived regarding the (in)ability of discrete dynamic programming to effectively tackle this problem within a rental context when realistically sized instances are considered.

Keywords Car rental · Dynamic programming · Dynamic pricing · Fleet deployment · Optimization model · Constraint programming

20.1 Introduction

This work deals with the integration of dynamic pricing decisions with resource capacity, deployment and consumption decisions within the car rental context. The goal is to decide, for the time horizon of a specific selling season:

B. B. Oliveira (✉) · M. A. Carravilla · J. F. Oliveira
INESC TEC and Faculty of Engineering, University of Porto, Porto, Portugal
e-mail: beatriz.oliveira@fe.up.pt

M. A. Carravilla
e-mail: mac@fe.up.pt

J. F. Oliveira
e-mail: jfo@fe.up.pt

- How many cars to have in the fleet,
- When to acquire them,
- How to deploy them among rental stations throughout the time horizon,
- How to assign them to rentals (that start and end throughout the time horizon and rental stations),
- How to price these rentals.

Car rental companies face a significantly price-sensitive demand. Since online sale channels have allowed companies to change their prices virtually instantaneously and with no cost, dynamic pricing is becoming a critical demand-management tool, not only in this sector but also in airlines, hotels and other businesses that rely on revenue management techniques (including pricing) to seize price-sensitivity and other demand segmentation characteristics.

In car rental, unlike the above-mentioned (more traditionally studied) sectors, the fleet is highly flexible and mobile, since the vehicles (resources) are easy(ier) to transfer, deploy and acquire. However, there is a myriad of products—the rentals—that share the same fleet capacity. The rentals are broadly characterized by their start and end date and location. Other elements (such as vehicle group required, for example) may characterize a type of rental. Nonetheless, for the sake of simplicity and clarity, throughout this work the fleet is assumed to be homogeneous and the pricing decisions, made for each rental type on its broader definition, can be considered as "reference prices" to which others are indexed (e.g. variations according to antecedence of purchase or extra conditions required). For a more detailed review on car rental fleet and revenue management works, the reader may refer to [5].

Recently, some interesting works have been dealing with the integration of dynamic pricing with capacity and inventory decisions [1, 7]. This integration has been becoming relevant for companies that can change and update their pricing policies and inventory and capacity decisions in an increasingly easier way, due to the improvement of the above-mentioned technological systems. The methodological approach applied often involves dynamic programming due to its affinity with the problem. Also, for the stochastic problem, other non-parametric approaches such as robust optimization have been developed. For a thorough and relevant review regarding dynamic pricing, especially when learning processes regarding demand are considered, the reader should refer to [2].

The work herein presented aims to tackle a similar problem, which differs on the type of capacity/inventory decisions made. In previously studied cases, the capacity/inventory was decided and considered to be available at the start of the horizon (or at multiple points throughout the horizon, through multiple capacity decisions) and then depleted by the demand until the end of the horizon. In the car rental (actually any rental) context, the capacity is not only affected by these decisions but also by "returning" (re-usable) resources. That is to say, the resource is not depleted by demand but only temporarily occupied and it will become available capacity again, possibly at a different location. This difference has a significant impact on the structure of the problem and motivated the research presented in this paper.

The goal of this work is to study the possibility to develop a solution method based on one of the most applied methodologies in the similar problem presented above—dynamic programming—and understand its advantages, limitations and drawbacks in this context. Besides its common application within the dynamic pricing and revenue management setting, dynamic programming has also been used on works that deal with car rental operational and fleet management problems, such as fleet size and deployment [4].

In this work, a general discrete dynamic programming framework is developed as well as two approaches to calculate the value of the decisions at each stage and state, which are presented in Sect. 20.2. Then, in Sect. 20.3, some illustrative numeric examples are used to analyze the limitations and advantages of the method. Finally, in Sect. 20.4, some conclusions are drawn.

20.2 Discrete Dynamic Programming Formulation

One important characteristic of this problem is that most decisions are intrinsically integer, such as the number of vehicles to acquire and deploy or the number of fulfilled rentals. Due to the detail considered for rental types, which aggregate rentals that start and end at specific locations and times, the order of magnitude of these decisions is relatively small and an approximate result obtained by relaxing the integrality constraints might be significantly impaired. Therefore, a discrete formulation was developed.

Dynamic programming provides a general framework to solve different problems, where a multistage structure is latent and can be used to decompose a complex problem into simpler sub-problems. Within this context, a *stage* represents a point where decisions are made. The goal is to formulate the problem so that at any stage the only information needed to make decisions is summarized on one or more *state* variables. The state at a specific stage is fully defined (on the deterministic case herein considered) by the state and decisions of the previous state, translated on a *state transition function*. At each stage, an *optimal-value function* can be defined, dependent on the current state and on the decisions made. Dynamic programming is then based on the recursive computation of the optimal-value function [3].

In this section, the stages, state variables and spaces, and transition functions will be defined. Also, two approaches to calculate the optimal-value function will be presented.

The notation for indexes and sets used throughout this paper is as follows:

n Index for stage;
e Index for state;
r Index for type of rental;
s, c Indexes for rental station;
p Index for price level;
$\mathscr{E}^n$ Set of states possible to achieve in stage n;

$\mathscr{S}$ Set of rental stations;
$\mathscr{R}$ Set of types of rental;
$\mathscr{R}_n^{\text{start}}$ Set of types of rentals that start at stage n;
$\mathscr{R}_{n,s}^{\text{start}}$ Set of types rentals that start at station s at stage n;
$\mathscr{P}$ Set of possible price levels.

Also, the following parameters will be considered:

T Number of time periods on the time horizon;
HC_n Holding cost for the fleet of vehicles existent at stage n (cost per vehicle);
TT_{sc} Empty transfer time between station $s \in \mathscr{S}$ and station $c \in \mathscr{S}$;
TC_{scn} Cost of initiating an empty transfer from station $s \in \mathscr{S}$ to station $c \in \mathscr{S}$ at stage n (cost per vehicle);
$DEM_r(q_r)$ Demand for type of rental $r \in \mathscr{R}$, dependent on the price q_r that is charged for this type of rental;
DEM_{rp} Demand for type of rental $r \in \mathscr{R}$ if price level $p \in \mathscr{P}$ is charged for this type of rental (alternative notation);
PRI_p Monetary value associated with price level $p \in \mathscr{P}$.

20.2.1 Stages

In the car rental context, the start and end dates that characterize the rental types can be aggregated in e.g. weeks. The same unit can be used for the capacity decisions due to the flexibility of some vehicle acquisition modes, such as leasing. These time units mark the decision points throughout the time horizon and are the most notorious element that contributes to the multistage structure of the problem.

The computation will follow the *backward induction* method, as it will start at the last time period and end at the first. That is to say, the calculation will start at $n = 0$, where n defines the number of stages missing to end the time period, and end at $n = T$.

The decisions made at each stage n are represented by the following variables:

u_r^n Number of rentals of type $r \in \mathscr{R}_n^{\text{start}}$ fulfilled at stage n;
q_r^n Price charged for rentals of type $r \in \mathscr{R}_n^{\text{start}}$;
w_s^n Number of vehicles acquired to be available in rental station $s \in \mathscr{S}$ at stage n;
y_{sc}^n Number of vehicles to deploy from station $s \in \mathscr{S}$ to station $c \in \mathscr{S}$ by an empty transfer that starts at stage n.

20.2.2 State Variables, Transition Function and State Spaces

At any stage, the state variables should provide all the information required to make the previously mentioned decisions. Dynamic formulations for inventory problems and integrated pricing and capacity problems use the stock or inventory available at each stage as the state variables.

In this case, in order to decide on number of rentals fulfilled (u-type decision), two types of information are required: the amount of demand for this type of rental, which is solely dependent on the pricing decision made at the same stage, and the stock of vehicles of each rental type available at the starting station, which depends on decisions from previous periods and should thus be summarized on the state variables.

At each station $s \in \mathscr{S}$ and stage n, this stock depends on the previous stock, the vehicles that leave the station (either occupied by rentals or transfers) and the vehicles that arrive (either at the end of rentals or transfers or by their acquisition) and can be computed by the following equation:

$$\begin{aligned}\text{stock}_s^n &= \text{stock}_s^{n+1} - \text{rentals that leave}_s^n - \text{transfers that leave}_s^n \\ &\quad + \text{rentals that arrive}_s^n + \text{transfers that arrive}_s^n + \text{vehicles aquired}_s^n \qquad (20.1)\end{aligned}$$

As previously discussed, since the state variables fully define the state, the transition function should only depend on the previous state. However, since the rentals and empty transfers may last for more than one time period, Eq. 20.1 requires information from stages other than the immediately subsequent.

Therefore, an artifact was developed and a second type of state variable introduced to represent the capacity occupied by current rentals or transfers that will be available in a later period. Figures 20.1 and 20.2 present an example to better explain these variables. As exemplified with the solid arrow, if there is a rental type that starts in $t = 1$ (station B) and ends in $t = 3$ (station A), the arrival of these rentals will affect the stock of vehicles available at station A in $t = 3$ (Fig. 20.1). However, this decision occurs on a stage other than the immediately consequent. With an additional stock variable, it is possible to memorize, for any stage, how many vehicles are currently occupied and will be available in a certain station in a certain number of time periods. In the example presented, as shown by the dashed arrows in Fig. 20.2, in $t = 2$, rentals of this type will increase the stock of vehicles currently occupied that will be available in station A in the next period. Then, in $t = 3$, the stock in station A will be increased by these units.

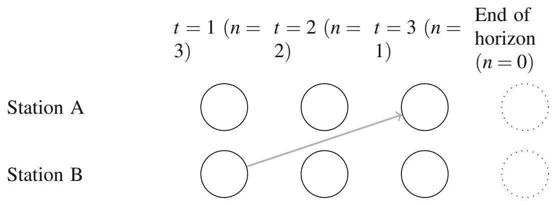
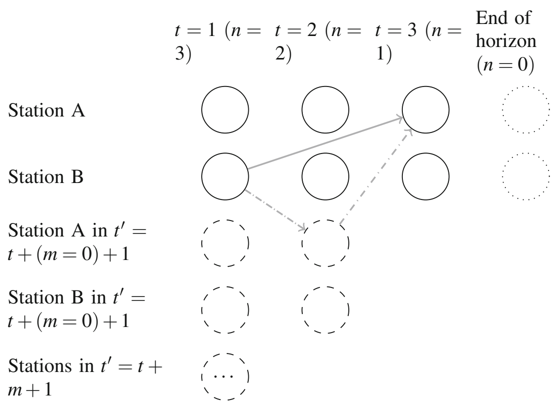


Fig. 20.1 Space-time network of an explanatory example, with 2 rental stations and 3 time periods. The arrow represents a rental type that starts in station B in $t = 1$ and ends in station A in $t = 3$



$t = 1$ ($n = 3$)
$t = 2$ ($n = 2$)
$t = 3$ ($n = 1$)
End of horizon ($n = 0$)
Station A
Station B
Station A in $t' = t + (m = 0) + 1$
Station B in $t' = t + (m = 0) + 1$
Stations in $t' = t + m + 1$
...

Fig. 20.2 Space-time network of Fig. 20.1, extended to include the representation of the additional state variables. The solid arrow is now decomposed in two dashed arrows using the additional state variable so that the state transition depends solely on the previous period

For each stage, the state variables can thus be defined as:

x_s^n Number of vehicles available in station $s \in \mathscr{S}$ (stock), at stage n;
o_{sm}^n Number of vehicles that are currently occupied and will be available in station $s \in \mathscr{S}$, at stage $n + m + 1$.

Thus, at each stage n, the state transition function t^n takes the following form:

$$\text{state}^{n-1} = t^n(u_r^n, q_r^n, w_s^n, y_{sc}^n, \text{state}^n) \tag{20.2}$$

$$= \begin{cases} x_s^{n-1} = x_s^n - \sum\limits_{r \in \mathscr{R}_{s,n}^{\text{start}}} u_r^n - \sum\limits_{c \in \mathscr{S}} y_{sc}^n + o_{s0}^n + w_s^n, \quad \forall s \in \mathscr{S} \\ o_{sm}^{n-1} = o_{s,m+1}^n + \sum\limits_{r \in \mathscr{R}_n^{\text{start}} \cap \mathscr{R}_{n+m+1,s}^{\text{end}}} u_r^n + \sum\limits_{c: c \in \mathscr{S}, TT_{cs} = m+2} y_{cs}^n, \\ \qquad \forall s \in \mathscr{S}, m = \{0, \ldots, n-2\} \end{cases}$$

State Space:
As for the state space, it was assumed that at the beginning of the time horizon ($n = T$), the state variables will be null (meaning that there are no vehicles occupied or in stock). For the remaining stages, an upper bound $XMAX$ must be defined for the x-type stock variables and another $OMAX$ for the o-type occupation variables. Each state is a combination of the possible values of these state variables. Therefore, the following equation defines the number of possible states NS, per stage $n < T$:

$$NS = \max[1, (OMAX + 1)^{|\mathscr{S}|(n-1)}] \times (XMAX + 1)^{|\mathscr{S}|} \tag{20.3}$$

Therefore, there are $NS \times |\mathscr{S}|$ x-type state variables per stage n and $NS \times |\mathscr{S}|(n-1)$ o-type state variables per stage $n > 1$.

20.2.3 Optimal-Value Calculation

At each stage n and state s_n, the maximum possible return over the n missing stages is given by the optimal-value function v_n. As previously discussed, this function v_n is defined recursively, based on the return f of this stage, which depends on the current decisions and state, and on the optimal-value function of the previous stage. Since the overall goal is to optimize the profit, the recursive optimization problem is broadly given by:

$$v^n(\text{state}^n) = \max \left\{ f^n(u_r^n, q_r^n, w_s^n, y_{sc}^n, \text{state}^n) + v^{n-1}\big(t^n(u_r^n, q_r^n, w_s^n, y_{sc}^n, \text{state}^n)\big) \right\}$$
$$\text{s.t. Constraints on decisions} \tag{20.4}$$

The return function f, in this case the profit obtained, is given by the difference between the revenue obtained from the price charged for each of the fulfilled rentals and the costs of holding the fleet of vehicles existent at this stage and the cost of initiating empty transfers:

$$f^n = \sum_{r \in \mathscr{R}_n^{\text{start}}} u_r^n \times q_r^n - z^n \times HC_n - \sum_{s \in \mathscr{S}} \sum_{c \in \mathscr{S}} y_{sc}^n \times TC_{scn} \tag{20.5}$$

The auxiliary decision variable z summarizes the total fleet and is defined, at each stage, by the sum of the vehicles acquired (decision variable), the vehicles in stock (state variable) and the vehicles occupied on rentals or transfers (state variable):

$$z^n = \sum_{s \in \mathscr{S}} \left(w_s^n + x_s^n + \sum_{m=0}^{n-1} o_{sm}^n \right) \tag{20.6}$$

The constraints on decisions are as follows:

- The price-dependent demand $(DEM_r(q_r))$ is an upper bound on the number of rentals that can be fulfilled:

$$u_r^n \leq DEM_r(q_r), \quad \forall r \in \mathscr{R}_n^{\text{start}} \tag{20.7}$$

- The overall capacity in a station limits the rentals fulfilled and the empty transfers leaving the station:

$$\sum_{r \in \mathscr{R}_{n,s}^{start}} u_r^n + \sum_{c \in \mathscr{S}} y_{sc}^n \leq x_s^n + w_s^n, \quad \forall s \in \mathscr{S} \tag{20.8}$$

- Also, an auxiliary constraint ensures that no empty transfers start and end in the same station:

$$y_{ss} = 0, \quad \forall s \in \mathscr{S} \tag{20.9}$$

- All decisions should lead to integer values.
- Additionally, the resulting state must be possible to achieve (i.e. be defined).

Two relevant characteristics of this optimization problem are the integrality of the decision variables and the non-linearity of the objective function. Therefore, two adequate optimization models and consequent resolution strategies were applied: a Mixed Integer Non Linear Program (MINLP) and a Constraint Programming (CP) model.

For each stage and state, the MINLP model proposed is a straightforward adaptation of the optimization problem summarized in (20.4) and (20.5). The main difference is related with the price decision variable q that is transformed into a binary variable q_{rp} that indicates whether or not a specific *price level* p from a previously defined set $\mathscr{P}$, which is associated with a monetary value PRI_p, is chosen for rental type r. This causes minor adaptations in the objective function and on the demand constraint (20.7). Also, binary variables st_e are added, to indicate whether or not the state achieved on the consequent stage from the decisions made will be state $e \in \mathscr{E}^{n-1}$ or not. This requires additional constraints to associate the binary variables with the consequent states and to ensure that at least one possible state is achieved. The model is thus as follows[1]:

$$
\begin{aligned}
\max f^n + v^{n-1} = {} & \left[\sum_{r \in \mathscr{R}_n^{\text{start}}} u_r^n \times \sum_{p \in \mathscr{P}} \left(q_{rp}^n \times PRI_p \right) - z^n \times HC_n - \sum_{s \in \mathscr{S}} \sum_{c \in \mathscr{S}} y_{sc}^n \times TC_{scn} \right] \\
& + \left[\sum_{e \in \mathscr{E}^{n-1}} st_e \times v^{n-1}(st_e) \right] \\
\text{s.t. } & (20.6), (20.8), (20.9) \\
& u_r^n \le DEM_{rp} + M(1 - q_{rp}), \quad \forall r \in \mathscr{R}_n^{\text{start}}, p \in \mathscr{P} \\
& [x_s^{n-1}]_{t^n} \le [x_s^{n-1}]_e + M(1 - st_e), \quad \forall e \in \mathscr{E}^{n-1}, s \in \mathscr{S} \\
& [x_s^{n-1}]_{t^n} \ge [x_s^{n-1}]_e - M(1 - st_e), \quad \forall e \in \mathscr{E}^{n-1}, s \in \mathscr{S} \\
& [o_{sm}^{n-1}]_{t^n} \le [o_{sm}^{n-1}]_e + M(1 - st_e), \quad \forall e \in \mathscr{E}^{n-1}, s \in \mathscr{S}, m \in \{0, ..., n-2\} \\
& [o_{sm}^{n-1}]_{t^n} \ge [o_{sm}^{n-1}]_e - M(1 - st_e), \quad \forall e \in \mathscr{E}^{n-1}, s \in \mathscr{S}, m \in \{0, ..., n-2\} \\
& \sum_{e \in \mathscr{E}^{n-1}} st_e \ge 1 \\
& u_r^n \in \mathscr{Z}_0^+, \forall r \in \mathscr{R}_n^{\text{start}}; \quad z^n \in \mathscr{Z}_0^+; \quad y_{sc}^n \in \mathscr{Z}_0^+, \forall s, c \in \mathscr{S} \\
& q_{rp}^n \in \{0, 1\}, \forall r \in \mathscr{R}_n^{\text{start}}, p \in \mathscr{P}; \quad st_e \in \{0, 1\}, \forall e \in \mathscr{E}^{n-1}
\end{aligned}
\tag{20.10}
$$

The second approach is based on Constraint Programming (CP), which aims to solve combinatorial problems by combining search and constraint solving, following

[1]The symbol $[\text{state}]_{t^n}$ indicates that the state expression was calculated based on the transition function and thus involves decision variables, while the symbol $[\text{state}]_e$ refers to an input/parameter: the state variables associated with state e.

the basis of logic programming [6]. This modeling and solving approach is suitable for integer, finite domain decision variables that are related by a set of constraints. Due to the characteristics of its associated search procedures, non-linearity presents no issue for CP models. Also, logic constraints such as “if-then-else” and implication statements can be implemented, which simplifies the model when compared with (20.10). For the sake of comparison between approaches, the price decision variable also refers to *price levels*, yet in this case it indicates directly the level, instead of having a binary variable per level. A similar reasoning is applied to the decision variable indicating the consequent state. The variable domains were considered as follows:

$$\begin{array}{ll}
u_r = \{0, \ldots, DUB_{s:\text{start}_r}\}, & \forall r \in \mathscr{R}_n^{\text{start}} \\
q_r \in \mathscr{P}, & \forall r \in \mathscr{R}_n^{\text{start}} \\
w_s = \{0, \ldots, DUB_s\}, & \forall s \in \mathscr{S} \\
y_{sc} = \{0, \ldots, x_s^n\}, & \forall s, c \in \mathscr{S} \\
z = \{0, \ldots, \sum_{s \in \mathscr{S}} DUB_s\} & \\
st \in \mathscr{E}^{n-1}, & \forall e \in \mathscr{E}^{n-1}
\end{array}$$

The demand upperbound per station DUB_s was calculated by:

$$DUB_s = \sum_{r \in \mathscr{R}_{n,s}^{\text{start}}} \left(\max_{p \in \mathscr{P}} DEM_{rp} \right) \tag{20.11}$$

The CP model is then similar to the previous one:

$$\begin{aligned}
\max f^n + v^{n-1} = & \left[\sum_{r \in \mathscr{R}_n^{\text{start}}} u_r^n \times q_r^n \times PRI_p - z^n \times HC_n - \sum_{s \in \mathscr{S}} \sum_{c \in \mathscr{S}} y_{sc}^n \times TC_{scn} \right] + \left[v^{n-1}(st) \right] \\
\text{s.t. } & (20.6), (20.8), (20.9) \\
& u_r^n \leq DEM_{rq_r}, \quad \forall r \in \mathscr{R}_n^{\text{start}} \\
& [x_s^{n-1}]_{t^n} = [x_s^{n-1}]_e \Rightarrow st = e, \quad \forall e \in \mathscr{E}^{n-1}, s \in \mathscr{S} \\
& [o_{sm}^{n-1}]_{t^n} = [o_{sm}^{n-1}]_e \Rightarrow st = e, \quad \forall e \in \mathscr{E}^{n-1}, s \in \mathscr{S}, m \in \{0, \ldots, n-2\}
\end{aligned} \tag{20.12}$$

Base Case:
To start the recursive calculation, it is important to define the base case for $n = 0$. Since in this problem it represents the end of the horizon, when no more rentals or vehicles are considered, it was assumed that $v^0 = 0$.

20.3 Illustrative Numeric Examples

Scope

The goal of this section is to provide some numerical examples that illustrate the drawbacks and limitations of this method in order to support the discussion on its adequacy. The main characteristics of the problem that influence will be identified to understand the potential and limits of its application.

From the discussion on the number of states and state variables, it was possible to verify that four main characteristics of the problem could significantly influence the effectiveness of the method proposed: the upper bound on the number of vehicles in stock in each station $XMAX$, the upper bound on the number of vehicles currently occupied to be available in a specific future period of time and station $OMAX$, the number of stations S and the number of time periods (i.e. stages).

From Eq. (20.3) it is possible to observe that the number of states in a stage easily explodes. Therefore, as an example, considering two rental stations and three periods of time: for $XMAX = 10$ and $OMAX = 5$,[2] the maximum number of states in a stage goes above 4.300, which leads to more than 17.000 state variables. If these numbers are doubled ($XMAX = 20$, $OMAX = 10$), the number of states becomes bigger than 53.000, with over 213.000 state variables.

The main issue is that this makes the effectiveness of the model highly dependent on two characteristics that are not intrinsic to the problem (although the maximum stock could have a clear parallel with the number of parking spaces available), and indirectly on the scale of the problem.

Data

Instances:
These numeric experiments are based on three cases that were adapted from instances provided for the Capacity-Pricing Problem in car rental,[3] which present a "photograph" of the rental system at a certain time, showing the demand for each type of rental, as well as the remaining parameters. The instances chosen were the ones where the number of rental types was (i) the smallest, (ii) the biggest and (iii) the median value. It is important to analyze how the approach performs varying this indicator since the number of rentals is one of the most relevant drivers of complexity to solve each sub-problem and, at the same time, it has virtually no impact on the number of states and stages, i.e. on the number of sub-problems to solve.

Experiment Environment:
The algorithms and MINLP and CP models were developed in C++/IBM ILOG Concert Technology and were run on a workstation computer with 48 Gigabyte of RAM memory, with 2 CPUs (Intel(R) Xeon(R) X5690 @ 3.46 GHz), with a 64-bit

[2]It is reasonable to assume that $OMAX \leq XMAX$.

[3]*Capacity-Pricing Model: car rental instances*, 2017, available at doi:http://dx.doi.org/10.17632/g49smv7nh8.1.

Operating System. The MINLP Solver used was CPLEX 12.6.1 and the CP solver used was CPLEX CP Optimizer 12.6.1.

Due to the number of stages and states, a time limit was set for calculating the optimal-value function. This makes it possible that the value obtained is not the optimum, yet it was considered as a mandatory control of the overall run time, since the MINLP or CP solver could experience difficulties in finding the optimum value or proving its optimality. The time limit chosen was 3 s. Preliminary experiments indicated that within this limit both solvers would often reach and prove the optimal result and that increasing it to 5 or 10 s would not significantly impact the results obtained. Nevertheless, it was considered that the possibility of no solution being found for a few specific stages and states was significant and impactful and therefore a "back-up" mechanism was developed so that in this case the time limit was slightly increased. Moreover, in the last stage (corresponding to the first time period), since only one state is possible, the time limit for its calculation was increased to 60 s in order to improve the probability of achieving and proving the optimum.

Results and Discussion

Figure 20.3 presents the best value obtained for each instance, with different combinations of the parameters $XMAX$ and $OMAX$. These numeric examples illustrate the potential and limitations of the approach proposed since they represent small configurations of the problem and already embody significant insights.

Firstly, if one compares the overall values obtained by both approaches, the one that uses the Constraint Programming model to calculate the optimal-value function (henceforward referred to as "CP approach" for the sake of brevity) obtains better

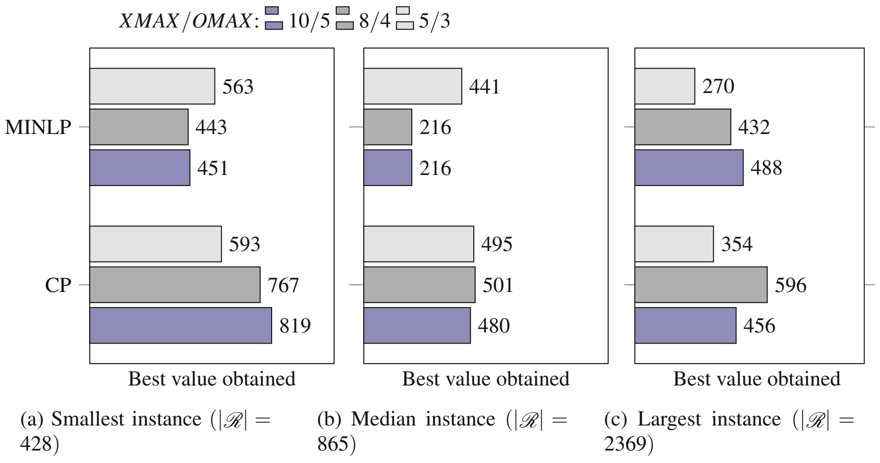


Fig. 20.3 Best profit values obtained by each approach, for each instance, with different combinations of $XMAX/OMAX$ parameters

results than the one that uses the Mixed Integer Non Linear Program ("MINLP approach"), especially for smaller instances.[4]

An interesting analysis can be made regarding the effect of the parameters $XMAX$ and $OMAX$ (directly connected with the number of states). It could be expected that an increase of these parameters would lead to higher values being obtained, since they represent a constraint on the "original problem" and their increase can be compared to a gradual relaxation of this constraint. Nevertheless, for the MINLP approach, this only happens for the biggest instance. In the remaining instances, increasing these parameters leads to a lower value. This might be explained by the effect of the time limitation imposed to each sub-problem. Due to this limit, the solver may not reach the optimum solution. Increasing the parameters makes the problem more complex to solve and thus makes the optimum more difficult to achieve. In fact, when the parameters are increased, the number of states increases and the sub-problems (which have decision variables and constraints dependent on the number of states) get more complex to solve.

As for the CP approach, a similar tendency is not as visible and it becomes difficult to draw conclusions regarding the relative strength of each of the two contradictory effects of increasing the parameters $XMAX$ and $OMAX$: (1) the "original problem" is more relaxed and thus better solutions could be achieved, and (2) the problems become more complex and, due to the time limit, the optimum value is more difficult to achieve.

From this analysis rises an interest in observing the actual run times to understand if the hypothesis related with the optimum being reached more easily is supported. Two bounds on expected time to solve can be drawn. The first is based on the number of optimization problems to solve and the time limit to solve them, not considering the previously mentioned extra-time rules. The second is the actual upper bound on time that considers all the extra-time rules. In order to reach the latter, the extra time would have to be used to its maximum for all optimization problems. Figure 20.4 presents the time to solve these numeric examples and reach the values presented in Fig. 20.3.

As expected, with an increase in instance size, there is a trend to increase the time to solve. Also, as it can be easily observed, the MINLP approach is consistently faster than the CP approach. In fact, the former is consistently below the expected bound (not considering extra time) while for the latter this only happens for the smallest instance. This means that the MINLP approach was often able to prove optimality in less than the time limit imposed, while the CP approach often used the extra time allowed. This does not fit in a straightforward way with the results previously discussed when comparing the best values obtained by each approach. In fact, the integrated analysis of Figs. 20.3 and 20.4 supports the claim that the CP approach quickly finds good solutions yet takes longer to reach the optimum (or prove optimality)

[4]Throughout this discussion, the notion of instance *size* will be associated with the intrinsic parameter being analyzed: the number of rental types. It thus especially related with the complexity of the optimization problems solved for each stage and state (not the number of stages and states *per se*).

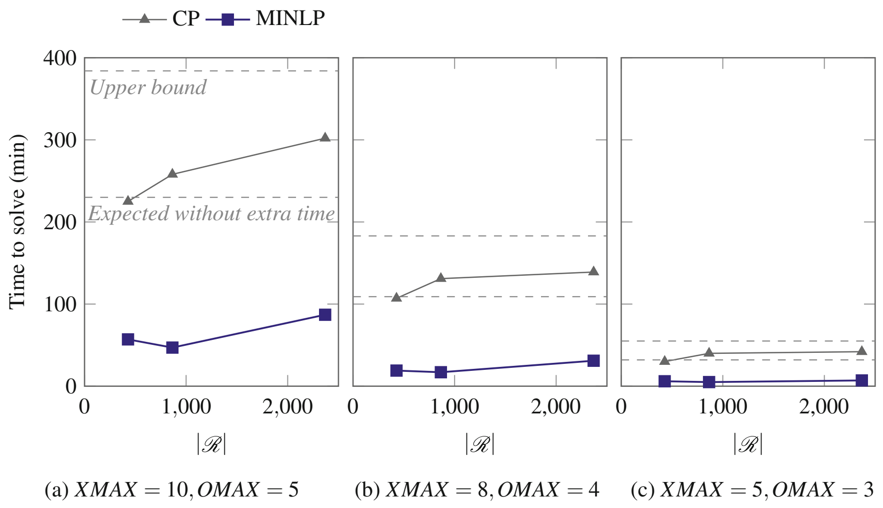


Fig. 20.4 Time to solve the numeric examples using the two approaches to calculate the optimal-value function, plotting the instances by the number of rental types $|\mathscr{R}|$, for three possible combinations of the parameters $XMAX$ and $OMAX$

Table 20.1 Comparison of key results and differences

	CP approach	MINLP approach
Overall best profit values	Generally higher	Generally lower
Effect of increasing size-influencing parameters	No significant effect on profit	Lower profit values
Time to solve	Significantly slower	Significantly faster
Conclusions	Quickly finds good solutions yet has difficulty proving optimality, increasing significantly the time to solve	Achieving/proving optimality ranges from extremely fast to impossible within time limit, making it difficult to obtain better solution values

and that the ability and speed to prove optimality varies significantly more in the MINLP approach: from extremely fast to prove optimality to returning a feasible solution with a high gap. This seems reasonable considering the characteristics of each solution method and the fact that the complexity of the optimization problems varies significantly among stages (within the same instance).

Table 20.1 summarizes and compares the key differences and results from the two approaches. Overall, it is possible to conclude that the time limit imposed has a significant impact. Nevertheless, although it can lead to poorer overall results, if a time limit is not imposed the time to solve could make both approaches nonviable.

20.4 Conclusions

In this work, a dynamic programming approach was developed to deal with the integrated dynamic pricing and capacity problem in the car rental business. This methodology has been successfully applied to similar problems and from the multi-stage structure of the problem (and the consequent "stock available" type of control) can be seen as an adequate method. Nevertheless, the fact that the capacity is "re-usable" in the rental context raises significant applicability issues that were analyzed.

The first drawback of applying dynamic programming to this context is that the number of states and state variables easily explodes. Already with these small numeric examples (in terms of number of time periods and rental stations, and considering deterministic demand) this method shows computational limitations. This is mainly due to the fact that the problem is related with a *rental* context—and this is why car rental is not like any other pricing and capacity/inventory model: the number of states explodes because stock can "return" after being depleted and that makes it necessary to keep track of occupied vehicles, which relates with decisions from time periods other than the immediately previous one.

An additional limitation is that the number of states is based on parameters that are not derived from the original problem, although they may have a close parallel to actual operational constraints, such as the stock of vehicles in a station being limited by the available parking spots. Although it was possible to observe that increasing the maximum number of vehicles in stock and occupied (and thus increasing the number of states) may hinder getting a better solution due to time limitations, not increasing these parameters for a real-world application of the methodology is not a viable option. In fact, the values herein proposed fail to fully reflect the reality of the problem. Ideally, these parameters should have no impact on the optimum value. Nevertheless, from a quick analysis of the order of magnitude of the demand values, it is easily established that in these numeric examples they have had impact.

These conclusions do not support the claim that dynamic programming is an adequate method to tackle this problem. Nevertheless, this discussion was able to bring some insights related with the problem structure as well as the potential and limitations of CP and MINLP when embedded in a discrete dynamic programming approach.

As future work, other methodologies will be applied to this rental context, especially considering the case of uncertain demand and realistically sized problems.

Acknowledgements The first author was supported by grant SFRH/BD/103362/2014 from FCT—Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology). This work was also partially financed by the ERDF—European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation—COMPETE 2020 Programme within project "POCI-01-0145-FEDER-006961", and by National Funds through the FCT—Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

References

1. E. Adida, G. Perakis, Dynamic pricing and inventory control: robust vs. stochastic uncertainty models-a computational study. Ann. Op. Res. **181**(1), 125–157 (2010). ISSN 02545330, https://doi.org/10.1007/s10479-010-0706-1
2. A.V. den Boer, Dynamic pricing and learning: historical origins, current research, and new directions. Surveys in Ope. Res. Manag. Sci. **20** (1), 1–18 (2015). ISSN 18767354, https://doi.org/10.1016/j.sorms.2015.03.001
3. A. Hax, S. Bradley, Dynamic programming (chapter 11), in *Applied Mathematical Programming*, vol. 26 (1977), pp. 320 – 362, https://doi.org/10.2307/2550876
4. Z. Li, F. Tao, On determining optimal fleet size and vehicle transfer policy for a car rental company. Comput. Op. Res. **37**(2), 341–350 (2010). ISSN 03050548, https://doi.org/10.1016/j.cor.2009.05.010
5. B.B. Oliveira, M.A. Carravilla, J.F. Oliveira, Fleet and revenue management in car rental companies: a literature review and an integrated conceptual framework. Omega **71**, 11–26 (2017). ISSN 03050483, https://doi.org/10.1016/j.omega.2016.08.011
6. F. Rossi, P. Van Beek, T. Walsh, *Handbook of Constraint Programming* (Elsevier, 2006). ISBN 0444527265
7. D. Simchi-Levi, X. Chen, J. Bramel, Integration of inventory and pricing (chapter 10), in *The logic of Logistics: Theory, Algorithms, and Applications for Logistics Management*, 3rd edn. (Springer Science+Business Media, Berlin, 2014), pp. 177–209