
Heuristics and Exact Algorithms for Choice-Based Capacitated and Uncapacitated Continuous Pricing

Tom Haering ^{*†} Robin Legault [‡] Fabian Torres ^{*}
Michel Bierlaire ^{*}

September 5, 2024

Report TRANSP-OR 240905
Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
Ecole Polytechnique Fédérale de Lausanne
`transp-or.epfl.ch`

^{*}École Polytechnique Fédérale de Lausanne (EPFL), School of Architecture, Civil and Environmental Engineering (ENAC), Transport and Mobility Laboratory, Switzerland, {tom.haering, fabian.torres, michel.bierlaire}@epfl.ch

[†]Corresponding author.

[‡]Massachusetts Institute of Technology (MIT), Operations Research Center, USA, legault@mit.edu

Abstract

We introduce the Breakpoint Heuristic Algorithm (BHA) to address the choice-based pricing problem (CPP) with and without capacity constraints as well as valid inequalities for the QCQP-L (quadratically constrained quadratic program with linear objective) formulation of the uncapacitated CPP, allowing us to speed up the exact Branch & Benders Decomposition (B&BD) approach. The BHA is based on the Breakpoint Exact Algorithm (BEA) which we extend to handle capacity constraints, implemented using an exogenous priority queue. Results show that, when optimizing the prices of two capacitated alternatives simultaneously, the BEA reports runtimes up to 20 times faster than the state-of-the-art mixed-integer linear programming (MILP) approach, while the BHA, leveraging a coordinate descent method, performs up to 5000 times faster than the MILP across low- and high-dimensional instances, while producing high-quality solutions. For the uncapacitated case, the BHA outpaces the BEA as well as the B&BD approach by multiple orders of magnitude, especially for high-dimensional instances. Our results further show significant improvements in computational time for the exact method (B&BD) when adding the valid inequalities together with the heuristic solution to guide the algorithm. The BHA significantly surpasses state-of-the-art methods, including those tailored for mixed logit models, in speed while maintaining an average optimality gap of less than 0.02%.

Keywords: discrete choice, pricing, capacity constraints, heuristic, valid inequalities

Glossary

Acronym	Full term
CPP	Choice-based Pricing Problem
DCM	Discrete Choice Model
ML	Mixed Logit
MILP	Mixed Integer Linear Programming / Program
QCQP-L	Quadratically Constrained Quadratic Program with Linear objective
BEA	Breakpoint Exact Algorithm
BHA	Breakpoint Heuristic Algorithm
ILS	Iterated Local Search heuristic
B&B	spatial Branch and Bound
B&BD	spatial Branch and Benders Decomposition
CoBiT	Convexification of a Biconvex optimization and Trust-region algorithm
LAG	Lagrangian decomposition-based heuristic

1 Introduction

Effective pricing strategies play a critical role in various industries, especially in markets where customer preferences and product choices are diverse and complex. Discrete choice

models (DCMs), based on random utility theory, provide a robust framework for capturing the impact of customer heterogeneity on demand. By modeling how individuals select among available alternatives, these models allow companies to better predict demand and optimize their pricing strategies accordingly. Building on this foundation, we aim to address the challenge of developing efficient algorithms for pricing with disaggregate demand modeling in the form of a DCM, which we will refer to as the choice-based pricing problem (CPP). Our central research question is how to create a scalable and computationally efficient algorithm to solve the CPP, especially for high-dimensional instances with customer heterogeneity and when taking constraints on capacity into account.

By advancing state-of-the-art methods, we seek to provide more effective solutions for real-world pricing challenges, where customer preferences and demand patterns can be modeled using random utility maximization (RUM) frameworks, resulting in so-called choice-based optimization problems. Utility maximization theory posits that individuals assess each option available to them and choose the one that maximizes their utility. Key areas where these models are applied include pricing Davis et al. (2017); Gallego & Wang (2014); Li et al. (2019); Paneque et al. (2022) and assortment optimization Liu et al. (2020); Rusmevichientong et al. (2010), which are fundamental to strategic business decisions. DCMs like logit and nested logit are a widely used tool to model demand in various problem settings, including facility location (Mai & Lodi, 2020; Ljubić & Moreno, 2018), railway timetabling (Cordone & Redaelli, 2011; Robenek et al., 2018), and toll setting (Wu et al., 2012) problems, as well as revenue management and pricing (Shen & Su, 2007; Korfmann, 2018; Gallego & Wang, 2014; Müller et al., 2021). For the latter, however, it has been shown that the more complex mixed logit (ML) DCMs represent the more powerful and realistic demand representation (Sumida et al., 2021; van de Geer & den Boer, 2022; Marandi & Lurkin, 2023). Incorporating advanced discrete choice models such as mixed logit into optimization problems introduces significant computational difficulties due to the non-convex nature of the choice probabilities (Hanson & Martin, 1996) and the need to represent individual demand perspectives, such as capacity allocation. As a result, only small to moderate-sized problems are typically solvable to optimality (Benati & Hansen, 2002; Paneque et al., 2021).

To tackle these computational challenges, innovative approaches have emerged: Gilbert et al. (2014) developed a tractable approximation for maximizing revenue through pricing under mixed logit (ML) demand in congested networks, utilizing a two-step process combining a mixed integer program with an ascent algorithm. Subsequently, Li et al. (2019) explored a price optimization problem under discrete ML demand, introducing a pair of concave maximization problems to bound the revenue function.

It is important to highlight that in Li et al. (2019) and van de Geer & den Boer (2022), both of which are closely related to our work, the probability measure is considered discrete. In Li et al. (2019), customer-specific variables are not included in the utility models, resulting in identical choice probabilities for all customers. In contrast, van de Geer & den Boer (2022) connects only the exogenous component of the utility to the customers. Consequently, unlike van de Geer & den Boer (2022), more recent studies, as well as our framework, address customer heterogeneity, including variations in price sensitiv-

ity parameters. For instance, Marandi & Lurkin (2023) consider discrete ML pricing with heterogeneous price-sensitivity parameters and propose an iterative optimization algorithm that asymptotically converges to the optimal solution. By formulating a linear optimization problem based on the trust-region approach, they find a feasible solution and design a convex optimization problem using a convexification technique to approximate the optimization problem from above. A branching method is then used to tighten the optimality gap. Results show that for various tested instances (up to 5 customers and 5 alternatives), their method significantly outperforms other approaches, among which van de Geer & den Boer (2022), in almost all cases. They furthermore successfully demonstrate the benefits in terms of expected revenue when going from MNL to ML demand modeling, a finding later confirmed by for example Abdolhamidi & Lurkin (2024)

One way to deal with the non-convexity arising from advanced DCMs is simulation-based optimization (Gosavi, 2015). This framework involves optimizing stochastic simulations to derive effective decisions or strategies. For instance, in choice-based optimization, this method approximates demand for products or services by simulating various random choices based on utility functions and using a sample average approximation method Haase & Müller (2013); Legault & Frejinger (2024).

To provide a more general framework for integrating advanced choice models into optimization problems, Paneque et al. (2021) proposed a simulation-based approach to formulate any choice-based optimization problem as a mixed-integer linear program (MILP). While increasing complexity due to the exponential scaling of the MILP solve time with the number of draws, this approach guarantees convergence to globally optimal solutions for sufficiently large numbers of draws. However, its practical applicability is limited to small-scale instances, highlighting the need for more efficient computational strategies. The authors manage to solve instances with two controlled prices, 50 customers and 250 simulation draws to optimality within two hours for the uncapacitated case and within 21 hours when taking capacity into account.

Recognizing this, Paneque et al. (2022) introduced a heuristic method based on a Lagrangian decomposition scheme, where the original problem is simplified by grouping simulation scenarios based on their similarities, allowing for a more efficient and scalable approach to solving the aforementioned MILP formulation. However, the efficiency of the algorithm depends heavily on the scenario grouping strategy, and its effectiveness might vary across different setups or datasets. They use the same data set as Paneque et al. (2021) and focus on instances with capacity constraints. For two controlled alternatives and 50 individuals, they manage to solve instances with 100 draws up to around 3% optimality within a two-hour time limit. For four controlled alternatives they solve instances with 25 draws in the same time limit to around 3.5% optimality. It is worth mentioning, however, that adding capacity constraints drastically complicates the optimization problem.

Haering et al. (2023) introduce two exact methods, the Breakpoint Exact Algorithm (BEA), well suited for low-dimensional instances with two controlled products or fewer, as well as a Spatial Branch and Benders Decomposition (B&BD) approach, performing well for high-dimensional instances, to tackle the uncapacitated pricing problem where demand is captured by any discrete choice model. They manage to outspeed

previous approaches by several orders of magnitude on low-dimensional instances and significantly reduce computational time for higher dimensions when using the B&BD algorithm. With the BEA, they solve instances with one controlled price, 50 customers and one million simulation draws within less than two minutes to optimality, as well as instances with two controlled prices, 50 customers and two thousand draws within two hours and as well as instances with the same number of controlled prices and customers but with 22 thousand draws within 72 hours. With the B&BD algorithm, they manage to solve instances with four controlled prices, 50 customers and 200 draws to 1% optimality within 24 hours. They furthermore compare their general method of integrating any DCM in an uncapacitated pricing problem with algorithms specifically designed for ML integration, and show that the BEA outperforms the work from Marandi & Lurkin (2023) by a factor of 300x for two-price optimization and an average factor of 3x for larger numbers of alternatives when using the spatial Branch and Bound (B&B) without Benders decomposition. However, differences in the approximation procedure have to be considered. These results show that the BEA and B&B / B&BD approaches represent the state-of-the-art solution methods for choice-based uncapacitated pricing and that for high-dimensional instances, the problem remains difficult to solve. Additionally, these approaches are not capable of handling capacity constraints.

To summarize, we find that for the CPP, there are no solution approaches in the literature that are at the same time general, flexible, and capable of solving realistic instances in a reasonable amount of time, especially for high numbers of prices and when constraints on capacity are introduced. Our aim is to address this gap by building on the work of Haering et al. (2023), specifically by extending the BEA algorithm to handle capacity constraints, developing an efficient and flexible heuristic for high-dimensional problems, and enhancing the capabilities of the B&B and B&BD procedures by adding valid inequalities. The goal in this context is to further demonstrate that a simulation-based method with a larger number of samples can be solved more efficiently than an approach that directly employs nonlinear choice functions but is restricted to a smaller number of classes or breaking points, as shown in Legault & Frejinger (2024). This provides a strong motivation for using a purely simulation-based approach with deterministic customers rather than approximating ML models with a smaller sample of customers following a logit model. By using a larger number of samples, the simulation-based method can more accurately capture the diversity of customer preferences and behaviors, thereby enhancing the robustness of the results and providing a more comprehensive understanding of complex customer heterogeneity.

We introduce the breakpoint heuristic algorithm (BHA) that can handle both the non-capacitated and capacitated version of the CPP and succeeds at finding very high-quality solutions for high-dimensional instances fast. It is based on the BEA, which we extend to handle capacitated instances. Furthermore, we show how the solution from the BHA can be used to speed up the exact spatial B&BD approach for the non-capacitated problem by guiding the pruning and introducing valid inequalities for the QCQP-L formulation of the problem, which the B&BD is based on.

The paper is structured as follows: Section 2 describes the choice-based pricing problem and its formulation as a mathematical program. In Section 3, we introduce the Breakpoint Heuristic Algorithm (BHA), as well as an extension with an iterated local search (ILS) to escape local optima. In Section 4 we demonstrate how to guide the B&BD algorithm using the solution from the BHA together with new valid inequalities, followed by Section 5 presenting the computational experiments. Finally, we conclude the paper and present its essential takeaways in Section 6.

2 Problem definition

Consider a competitive market with multiple products, of which J products are controlled by a supplier who wants to identify the set of prices that maximizes their revenue. We number the controlled alternatives from 1 to J , and the competitors' alternatives using non-positive numbers, from $1 - K$ to 0. Denote $C^1 = \{1, \dots, J\}$ the set of all offered alternatives by the supplier and $C = C^1 \cup \{1 - K, \dots, 0\}$ the set of all alternatives available. We consider N customers choosing one product among all offered alternatives. Each individual $n \in \mathcal{N} = \{1, \dots, N\}$ may furthermore have a different set of considered alternatives, denoted by $C_n \subset C$. We need to assume that each individual has at least one uncontrolled alternative in the choice set. If not, the problem is unbounded. An individual's considered set of alternatives that are offered by the supplier is denoted by $C_n^1 = \{i \in C_n | i \in C^1\}$. The behavior of the customers is captured by a random utility model: each alternative $i \in C_n$ is associated with a stochastic utility U_{in} , which depends on socioeconomic characteristics of individual n , alternative-specific attributes, and the controlled prices for alternatives $i \geq 1$. It can be defined as follows:

$$\begin{aligned} U_{in} &= V_{in} + \varepsilon_{in} & \forall i \in C_n \setminus C_n^1, \\ U_{in} &= V_{in} + \beta_p^{\text{in}} p_i + \varepsilon_{in} & \forall i \in C_n^1, \end{aligned}$$

where V_{in} represents the deterministic part of the utility that is observed by the analyst, which can take any form and be non-linear in the explanatory variables, and ε_{in} is the unobserved error term (and thus a random variable). For the B&BD algorithm, the assumption needs to be made that the utilities are linear in the prices p_i , which are multiplied by a pricing coefficient $\beta_p^{\text{in}} < 0$, that can vary across n and i . Any assumption on the influence of the price on the utility can be dropped completely for the BEA, BEAC and BHA approaches, although behavioral realism suggests that the utility is at least non-increasing in the price. It is furthermore worth noting that for all approaches, including the B&BD, the price variable p_i can be separated into multiple p_{ic} for different customer segments c without significantly altering the algorithms. For the sake of readability, we adhere to the global price notation p_i . The probability $P_n(i)$ that individual n chooses alternative $i \in C_n$ can now be written as follows:

$$P_n(i) = \mathbb{P}(U_{in} \geq U_{jn} \forall j \in C_n) \quad \forall i \in C_n$$

The controlled prices p_i , $i \in C^1$ are decision variables that need to be optimized in order to maximize the expected revenue, expressed as each product's price times the probability

the product is bought by an individual, summed up over all individuals. We assume each price p_i to be bounded within a continuous domain $[p_i^L, p_i^U]$. In general, the mathematical expression for $P_n(i)$ is complex. Advanced models, such as mixtures of logit and hybrid choice models lack a closed form and are expressed using integrals (Hanson & Martin, 1996).

2.1 Problem formulation

To address the lack of closed-form expressions for the probability functions, we employ the simulation approach of Paneque et al. (2021): We take R draws ε_{inr} from the distribution of the error terms to generate R scenarios (the terms ‘‘scenario’’ and ‘‘draw’’ will henceforth be used interchangeably) with deterministic utilities U_{inr} :

$$\begin{aligned} U_{inr} &= V_{in} + \beta_p^{in} p_i + \varepsilon_{inr} & \forall i \in C^1, n \in \mathcal{N}, r \in \mathcal{R}, \\ &= c_{inr} + \beta_p^{in} p_i & \forall i \in C^1, n \in \mathcal{N}, r \in \mathcal{R}, \\ U_{inr} &= V_{in} + \varepsilon_{inr} & \forall i \in C \setminus C^1, n \in \mathcal{N}, r \in \mathcal{R}, \\ &= c_{inr} & \forall i \in C \setminus C^1, n \in \mathcal{N}, r \in \mathcal{R}, \end{aligned}$$

where $\mathcal{R} = \{1, \dots, R\}$ and c_{inr} contains all terms of the utility function independent from the price. As now all uncontrolled alternatives $i \in C \setminus C^1$ have a utility that is constant given an individual n and a scenario r , we can gather them for each tuple (n, r) as a single opt-out alternative, corresponding to the best of them:

$$c_{0nr} = \max_{i \in C \setminus C^1} c_{inr} \quad \forall n \in \mathcal{N}, r \in \mathcal{R}.$$

We thus redefine $C = C^1 \cup \{0\}$ and impose $0 \in C_n \forall n \in \mathcal{N}$, as otherwise, the problem is unbounded. The choice of individual n in scenario r is then modeled with the choice variable ω_{inr} , which is equal to 1 if alternative i is chosen and 0 otherwise. Subsequently, the probability of an individual $n \in \mathcal{N}$ choosing alternative $i \in C_n$ can be approximated by the sample average $\frac{1}{R} \sum_{r \in \mathcal{R}} \omega_{inr}$.

This framework leads to the formulation of the uncapacitated choice-based pricing problem (CPP) as a quadratically constrained quadratic program with linear objective (QCQP-L), presented in Haering et al. (2023), given in Formulation 1.

$$\begin{aligned}
& \max_{p, \omega, \eta, \mathbf{U}, \mathbf{h}} \frac{1}{R} \sum_{r \in \mathcal{R}} \sum_{n \in \mathcal{N}} \sum_{i \in C_n^1} \eta_{inr} \\
& \text{s.t.} \\
& \sum_{i \in C_n} \omega_{inr} = 1 \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\mu_{nr}) \\
& h_{nr} = c_{0nr} \omega_{0nr} \\
& \quad + \sum_{i \in C_n^1} [c_{inr} \omega_{inr} + \beta_p^{in} \eta_{inr}] \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\zeta_{nr}) \\
& h_{nr} \geq c_{0nr} \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\alpha_{0nr}) \quad (1) \\
& h_{nr} \geq U_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\alpha_{inr}) \\
& U_{inr} = c_{inr} + \beta_p^{in} p_i \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\kappa_{inr}) \\
& \eta_{inr} = p_i \omega_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}) \\
& \omega \in [0, 1]^{(J+1)NR} \\
& p \in [p_1^L, p_1^U] \times \dots \times [p_J^L, p_J^U] \\
& \eta \in [0, p_1^U] \times \dots \times [0, p_J^U] \\
& \mathbf{U}, \mathbf{h} \in \mathbb{R}^{JNR}, \mathbb{R}^{NR}
\end{aligned}$$

The objective function is equal to the expected revenue and is thus defined as the approximated choice probability of individual n selecting alternative $i \in C_n$ multiplied by the alternative's price p_i , summed over all individuals. The constraints define the individual choices: Constraints (μ_{nr}) guarantee that exactly one alternative is chosen per individual and scenario. Constraints (κ_{inr}) model the utility U_{inr} of each alternative i for individual n in scenario r . Constraints (ζ_{nr}) and constraints (α_{inr}) enforce the optimality conditions for the customer utility maximization problem. Note that this is a continuous, but non-convex, reformulation of the original mixed-integer linear program (MILP) approach. Indeed the previously integer ω variables are relaxed to be in $[0, 1]$ instead, at the cost of non-convex choice constraints. This non-convexity stems from multiplying two continuous variables in the price p_i and the choice ω_{inr} , isolated in the constraints (λ_{inr}) . Finally, the price variables may also depend on the individuals (or groups of individuals), thus allowing for segmented targeting of the population.

Formulation 1 can be extended to incorporate capacity constraints on the alternatives, as demonstrated in Paneque et al. (2021). We include this extension in Appendix A. It is furthermore worth noting that, if the prices are all fixed to constant values, it is trivial to find the optimal values of all variables. To see this, it is enough to observe that with fixed prices p , the problem reduces to solving a utility maximization knapsack problem for each customer n and scenario r . As the prices are the only connecting variables, this can be done separately for every tuple (n, r) , giving an efficient way to evaluate the objective function for a given feasible solution. A similar procedure can be applied to the capacitated version of the problem.

3 Methodology

In this section, we introduce the Breakpoint Heuristic Algorithm (BHA) which can be applied to solve both the uncapacitated and capacitated version of the CPP, as well as an iterated local search (ILS) heuristic to improve the solution quality. Lastly, we demonstrate how to guide exact methods using the heuristic result together with newly developed valid inequalities for the CPP.

3.1 Breakpoint heuristic algorithm (BHA)

The BHA is based on the exact Breakpoint Exact Algorithm (BEA), introduced in Haering et al. (2023). It capitalizes on the idea of decision-making breakpoints. Given a fixed state of the market—a set of controlled and competing alternatives, all with fixed prices—we can compute, for each customer and each simulation scenario, a breakpoint at which the utility of a newly introduced alternative becomes the largest, thus altering the current decision. These breakpoints represent a set of local optima that can be enumerated. For details to the BEA we refer the reader to Haering et al. (2023). The extension to handle capacity constraints is straightforward. When a new alternative is introduced into a fixed state, customers may switch from any existing option, not just the current best, as capacity constraints could force them to make suboptimal choices. Consequently, decision breakpoints between all current alternatives and the new one must be considered. Furthermore, for each combination, we invoke a function `compute_objective_value` that calculates the objective value for these prices, taking capacity restrictions into account, as due to interdependent choices potentially leading to recursive substitution, continuously updating choices and revenues (as in the original BEA) becomes impractical.

This extension of the BEA is discussed in more detail in Appendix B.

The BHA can be summarized as a coordinate descent (ascent), iteratively optimizing one price at a time while fixing all other prices, terminating once no coordinate can be improved further. It is described in the following procedure:

1. Choose a starting point for the heuristic. As any combination of prices is feasible, the simplest choice here can be to choose the middle of the price bounds, $p^* = (\frac{p_i^L + p_i^U}{2})_{i \in C^1}$.
2. Evaluate the objective function for price p^* , giving objective value o^* .
3. Set $j = 1$.
4. Solve the problem using the BEA but with modified bounds \bar{p}^L, \bar{p}^U , where, $\bar{p}_i^L = \bar{p}_i^U = p_i \forall i \neq j$ and $\bar{p}_j^L = p_j^L, \bar{p}_j^U = p_j^U$. We refer to these new bounds as $\bar{p}^L(p, j, p^L)$ and $\bar{p}^U(p, j, p^U)$. All bounds, except for alternative j , are thus tight, greatly simplifying the problem. We thus iterate over all relevant breakpoints for all simulated customers, evaluating the objective value at each combination of breakpoints and updating the highest objective o^* and the best prices p^* whenever a better solution is found.

5. Set $j = j + 1$ and repeat from step 4. In the case of $j = D$, we reset it to $j = 1$.
6. Terminate once no change in the optimal solution is observed over D iterations.

The pseudocode for the BHA is provided in Algorithm 1.

Algorithm 1: Breakpoint Heuristic Algorithm (BHA)

```

Function BHA ( $p_{start}; c, prio\_queue$ ) :
   $o^* \leftarrow compute\_objective\_value(p_{start})$ 
   $p^* \leftarrow p_{start}$ 
   $j \leftarrow 1$ 
   $\sigma \leftarrow 0$ 
   $i \leftarrow 0$ 
  while  $\sigma < D$  do
     $i += 1$ 
     $\lambda = false$ 
     $\hat{p}_j, \hat{o} \leftarrow BEA(\bar{p}^L(p, j, p^L), \bar{p}^U(p, j, p^U); c, prio\_queue)$ 
     $p_j \leftarrow \hat{p}_j$ 
    if  $\hat{o} > o^*$  then
       $o^* \leftarrow \hat{o}$ 
       $p^* \leftarrow p$ 
       $\sigma \leftarrow 0$ 
    else
       $\sigma += 1$ 
    end
     $j += 1$ 
    if  $j > D$  then
       $j \leftarrow 1$ 
    end
  end
  return  $o^*, p^*$ 
end

```

3.2 Iterated local search heuristic (ILS)

The iterated local search algorithm is an iterative enhancement to the BHA, aimed at escaping local optima through adaptive step size adjustments. Initiated with a set of initial prices p and an objective value o^* , the algorithm takes as additional inputs an initial step size δ , the number of steps to be taken in each direction (increase and decrease) k , a step increase factor γ , and a maximum step size Δ_{max} . Each iteration consists of the following steps:

- Exploring both increase and decrease directions for each price component $i \in C^1$, the algorithm tests k increment steps, each being a multiple of δ . At each step, a new candidate solution \bar{p} is generated by adjusting the i -th component of p by $\pm k\delta$.

- For each candidate solution, the objective function is evaluated, and o^* is updated when a new best solution is found.
- If there was no improvement in objective value after completing the line search on all components of p , the step size δ is increased by the factor γ , enlarging the scope of the line search in the next iteration.

The algorithm continues until $\delta \geq \Delta_{max}$. The pseudocode for the ILS can be found in Algorithm 2.

Algorithm 2: Iterated Local Search Algorithm (ILS)

Function

```

iterated_local_search( $p_{start}, \delta, k, \gamma, \Delta_{max}; caps, prio\_queue$ ):
   $o^* \leftarrow compute\_objective\_value(p_{start})$ 
   $p^* \leftarrow p_{start}$ 
   $\varphi \leftarrow true$ 
   $\sigma \leftarrow 0$ 
  while  $\delta < \Delta_{max}$  do
     $\varphi \leftarrow false$ 
    for  $j \in 1 : D$  do
      for  $d \in [-1, 1]$  do
        for  $l \in 1 : k$  do
           $p^{new} \leftarrow p$ 
           $p_j^{new} += d \cdot l \cdot \delta$ 
          if  $p_j^L \leq p_j^{new} \leq p_j^U$  then
             $o^{new}, p^{new} \leftarrow BHA(p^{new}; c, prio\_queue)$ 
            if  $o^{new} > o^*$  and  $p^{new} \in [p^L, p^U]$  then
               $o^* \leftarrow o^{new}$ 
               $p^* \leftarrow p^{new}$ 
               $\varphi \leftarrow true$ 
               $\sigma \leftarrow 0$ 
            end
          end
        end
      end
    end
    end
    if  $\varphi$  then
       $\sigma += 1$ 
       $\delta := \gamma$ 
    end
    else
       $\sigma \leftarrow 0$ 
    end
     $p \leftarrow p^*$ 
     $o \leftarrow o^*$ 
  end
  return  $o^*, p^*$ 
end

```

4 Guiding the exact algorithm

As shown in Section 5, the heuristics introduced above provide very good solutions in a short amount of time. However, they do not provide a guarantee of optimality. Therefore, we also investigate the possibility of solving the problem exactly. In this context, we use the heuristic to help an exact algorithm. We consider as a starting point the spatial Branch & Benders algorithm for the uncapacitated CPP introduced by Haering et al. (2023). We give a brief overview of the method and then explain how we can speed up its convergence. The main strategy for accelerating the performance of Branch and Bound algorithms hinges on improving the bounds. Specifically, using a heuristic such as the BHA to find an initial feasible solution generates a strong lower bound (for a maximization problem) on the objective, significantly reducing the number of nodes explored in the search tree by enabling more effective pruning. Additionally, the upper bounds, needed to prove optimality, can be improved through the incorporation of valid inequalities. Lastly, we incorporate knowledge of the heuristic solution into the enumeration strategy to break ties between nodes with equal upper bound.

4.1 Spatial Branch and Benders algorithm

The spatial Branch and Benders (B&BD) algorithm proposed by Haering et al. (2023) solves the mathematical program stated in Formulation 1 by first employing the McCormick envelope (McCormick, 1976) to relax the bilinear constraints defining η_{inr} , and then tightens that relaxation by finding the best set of bound via a spatial branch and bound tree. In each child node, the length of the interval between the price bounds for a selected price is halved, guaranteeing convergence. The relaxation in each node is then solved using Benders decomposition.

Given a set of bounds $p_i \in [p_i^L, p_i^U] \forall i \in C^1$, the McCormick envelop used to relax the constraints $\eta_{\text{inr}} = p_i \omega_{\text{inr}}$ is given by:

$$\begin{aligned} \eta_{\text{inr}} &\geq p_i^L \omega_{\text{inr}} \\ \eta_{\text{inr}} &\geq p_i^U \omega_{\text{inr}} + p_i - p_i^U \\ \eta_{\text{inr}} &\leq p_i^L \omega_{\text{inr}} + p_i - p_i^L \\ \eta_{\text{inr}} &\leq p_i^U \omega_{\text{inr}} \end{aligned}$$

This yields the linear McCormick relaxation of the QCQP-L shown in Formulation 2. Then, a spatial Branch and Bound algorithm, see for example Liberti (2008), is employed to find the globally optimal values for all the prices. A conceptual outline of the method is given below:

1. Solve the McCormick relaxation (Formulation 2) using the initial bounds.
2. From the solution value of the prices, compute the corresponding choices and construct a feasible solution.

3. Choose a price to branch on, then split the search interval for that price, i.e., its bounds, into two, while all other price bounds remain the same. Add two new nodes to the Branch and Bound tree, each corresponding to one set of the new bounds.
4. Choose the next node from the tree based on the achieved objective value in its parent node (*best-first-search*), and solve the relaxation with the bounds corresponding to that node.
5. Continue until the relative gap between the objective value of the tightest relaxation is close enough (up to a predefined relative optimality gap) to the objective value of the best feasible solution found.

In every node of the Branch and Bound tree, Formulation 2 needs to be solved with a given set of price bounds, which may be time-consuming due to a large number of variables η and ω . However, Formulation 2 is highly separable: indeed, if all variables p_i are fixed to a certain value, the utility maximization problem can be solved for every individual and scenario independently. This is why a Benders decomposition approach is considered to speed up the solution of the McCormick relaxation in each node of the Branch and Bound tree. Benders decomposition works by decomposing the original problem into a master problem and a subproblem, where the master problem is a relaxation of the original problem that iteratively is improved by the addition of optimality and feasibility cuts, see for example Rahmaniani et al. (2017). For more details on the B&BD algorithm, see Haering et al. (2023).

$$\begin{aligned}
& \max_{p, \omega, \eta, \mathbf{U}, h} \frac{1}{R} \sum_{r \in \mathcal{R}} \sum_{n \in \mathcal{N}} \sum_{i \in C_n^1} \eta_{inr} \\
& \text{s.t.} \\
& \sum_{i \in C_n} \omega_{inr} = 1 \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\mu_{nr}) \\
& h_{nr} = c_{0nr} \omega_{0nr} \\
& \quad + \sum_{i \in C_n^1} [c_{inr} \omega_{inr} + \beta_p^{in} \eta_{inr}] \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\zeta_{nr}) \\
& h_{nr} \geq c_{0nr} \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\alpha_{0nr}) \\
& h_{nr} \geq \mathbf{U}_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\alpha_{inr}) \quad (2) \\
& \mathbf{U}_{inr} = c_{inr} + \beta_p^{in} p_i \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\kappa_{inr}) \\
& \eta_{inr} \geq p_i^L \omega_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^1) \\
& \eta_{inr} \geq p_i^U \omega_{inr} + p_i - p_i^U \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^2) \\
& \eta_{inr} \leq p_i^L \omega_{inr} + p_i - p_i^L \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^3) \\
& \eta_{inr} \leq p_i^U \omega_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^4) \\
& \omega \in [0, 1]^{(J+1)NR} \\
& p \in [p_1^L, p_1^U] \times \dots \times [p_J^L, p_J^U] \\
& \eta, \mathbf{U}, h \in \mathbb{R}^{JNR}, \mathbb{R}^{JNR}, \mathbb{R}^{NR}
\end{aligned}$$

4.2 Valid inequalities

A first general set of valid inequalities can be derived by observing that, given a set of bounds $[p_i^L, p_i^U]_{i \in C^1}$, i.e. at each node in the tree, for each simulated customer $(n, r) \in \mathcal{N} \times \mathcal{R}$ and alternative $i \in C_n^1$, there exists a *minimal breakpoint* \check{p}_i^{nr} (assuming strongest competition) and a *maximal breakpoint* \hat{p}_i^{nr} (assuming weakest competition), defined as follows:

$$\begin{aligned}
\check{p}_i^{nr} &= \frac{\max_{j \in C_n^1 \setminus \{i\}} \mathbf{U}_{jnr}(p_j^L) - c_{inr}}{\beta_p^{in}} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \\
\hat{p}_i^{nr} &= \frac{\max_{j \in C_n^1 \setminus \{i\}} \mathbf{U}_{jnr}(p_j^U) - c_{inr}}{\beta_p^{in}} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}
\end{aligned}$$

For the strongest competition, we compute the maximum utility among all other controlled alternatives, given their lowest possible prices given the current bounds. As we assume $\beta_p < 0$ in all cases, this corresponds to the strongest possible competition. On the other hand, taking the maximum utility among all other controlled alternatives, given their highest possible prices corresponds to the breakpoint assuming the weakest competition.

These breakpoints exhibit the following properties:

$$\begin{aligned}
p_i \leq \check{p}_i^{nr} &\implies (n, r) \text{ is guaranteed to select } i \\
&\implies \omega_{inr} \geq 1 \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, \\
p_i \geq \hat{p}_i^{nr} &\implies (n, r) \text{ is guaranteed to } \textit{not} \text{ select } i \\
&\implies \omega_{inr} \leq 0, \eta_{inr} \leq 0 \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R},
\end{aligned}$$

which can be integrated into Formulation 1 by use of the following valid inequalities:

$$p_i + (\check{p}_i^{nr} - p_i^L)\omega_{inr} \geq \check{p}_i^{nr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R},$$

and

$$\begin{aligned}
p_i + (p_i^U - \hat{p}_i^{nr})\omega_{inr} &\leq p_i^U \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}, \\
p_i + (p_i^U - \hat{p}_i^{nr})\eta_{inr} &\leq p_i^U \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R},
\end{aligned}$$

respectively. Note that in case of the upper bound with \check{p}_i^{nr} , we cannot deduce a bound on the revenue η_{inr} except that it is less than or equal to \check{p}_i^{nr} which is already covered by the McCormick constraints.

Given a feasible solution p^* to the CPP, we can furthermore derive the following relations:

$$\begin{aligned}
p_j \geq p_j^* \quad \forall j \neq i, \quad p_i \leq p_i^* &\implies \omega_{inr} \geq \omega_{inr}^* \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \\
p_j \leq p_j^* \quad \forall j \neq i, \quad p_i \geq p_i^* &\implies \omega_{inr} \leq \omega_{inr}^* \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R}
\end{aligned}$$

We describe how to convert the first relation into linear constraints, as the second follows symmetrically. As the left-hand side consists of J individual conditions that all need to be verified, we introduce an auxiliary variable $z_{inr} \in [0, 1]$ for each such condition i and simulated customer (n, r) . Note that these variables are indicators for whether or not a condition holds, and as such are binary in nature. However, including them as binary variables would inevitably slow down the computation of the relaxation significantly, which is why we relax their domain. For each $i \in C^1$, we model the z variables with the following constraints:

Constraints for alternative i :

$$p_i + (p_i^* - p_i^L)z_{inr} \geq p_i^* \quad \forall n \in \mathcal{N}, r \in \mathcal{R}$$

Constraints for alternatives $j \neq i$:

$$p_j - (p_j^U - p_j^*)z_{jnr} \leq p_j^* \quad \forall n \in \mathcal{N}, r \in \mathcal{R}$$

Additionally, we add constraints to ensure the relationship between the z variables and the choice variables ω :

$$\omega_{inr} \geq \sum_{j=1}^J z_{jnr} - (J - 1) \quad \forall n \in \mathcal{N}, r \in \mathcal{R}$$

These constraints ensure that $\omega_{inr} \geq 1$ if all conditions hold, i.e., $z_{inr} = 1 \forall i \in C_n^1$. Although the relaxation of the z variables' domain makes these inequalities less tight, they remain valid, as each individual inequality still holds for fractional values of z . However, this change makes the computational gain of their addition to the relaxation dependent on the problem instance.

The valid inequalities developed in this subsection can directly be integrated in Formulation 2.

4.3 Improving price bounds

Given a set of price bounds $[p_i^L, p_i^U]_{i \in C^1}$ at a node in the branch and bound tree, we can look to improve the price bounds further before solving the relaxation. For this, we consider the lowest minimal breakpoint \check{p}_i and highest maximal breakpoint \hat{p}_i for each $i \in C^1$ over all individuals and scenarios:

$$\check{p}_i := \min_{n \in \mathcal{N}, r \in \mathcal{R}} \check{p}_i^{nr} \quad \forall i \in C^1$$

$$\hat{p}_i := \max_{n \in \mathcal{N}, r \in \mathcal{R}} \hat{p}_i^{nr} \quad \forall i \in C^1$$

From their derivation, we can infer certain conditions based on these values. If $p_i > \hat{p}_i$, this implies that no customer will choose alternative i . Conversely, if $p_i < \check{p}_i$, it indicates that every customer will choose alternative i , provided it is within their choice set.

Additionally, we can refine these bounds to aim for target-specific outcomes, if desired. For instance, if p_i exceeds the m -th highest maximal breakpoint $\hat{p}_{i,m}^{nr}$, it suggests that at most m simulated customers will choose alternative i . Similarly, if p_i is below the m -th lowest minimal breakpoint $\check{p}_{i,m}^{nr}$, it implies that at least m simulated customers will choose alternative i . In our case, we assume that for each product, there should be at least one customer or scenario in which the product is chosen. If this assumption does not hold, the product can be considered as having no effect on the choice and can be removed from the set of offered products entirely. Consequently, we update the upper bound p_i^U by replacing it with \hat{p}_i whenever $\hat{p}_i < p_i^U$, for all $i \in C^1$.

4.4 Node enumeration

Finally, when multiple nodes in the branch and bound tree have an identical upper bound, we can use the knowledge of a good solution to determine the next node to explore. Specifically, we select the node that contains the highest number of price values of the provided solution within its bounds, exploring promising areas first. Let Ω denote the set of all active nodes in the tree, where a node j consists of a set of bounds Δ^j and its upper bound on the objective value $\hat{\delta}_j$. Furthermore, denote p^* as the provided starting solution. The next node to be selected from the tree can now be written as:

$$j^* := \arg \max_{j \in \Omega} \{ |\{k \in C^1 \text{ s.t. } p_k^* \in [p_k^L(j), p_k^U(j)]\}| \text{ s.t. } \hat{\delta}_j = \max\{\hat{\delta}_i \mid \{\Delta^i, \hat{\delta}_i\} \in \Omega\} \}$$

implying that, among the nodes that reach the highest upper bound on their potential objective value, we choose the one that contains the highest number of price values of the initial solution in its bounds.

5 Results and discussion

In this section, we apply the introduced methodology to a parking choice case study, evaluating its performance across various instance sizes. We explore the limits of solvable instances, compare the results to state-of-the-art methods for choice-based pricing and mixed logit-specific pricing, and assess the computational efficiency and solution quality.

5.1 Case study

To test the presented methodology we rely on the same mixed logit (ML) case study as various other studies, among which Bortolomiol et al. (2021); Paneque et al. (2022); Marandi & Lurkin (2023); Haering et al. (2023), establishing itself as a popular benchmark data set for ML-based pricing policies. The case study concerns a parking services operator, motivated by the published disaggregate demand model for parking choice by Ibeas et al. (2014). The choice set consists of three services: paid on-street parking (PSP), paid parking in an underground car park (PUP), and free on-street parking (FSP), presenting the opt-out. We artificially add more PSP or PUP options by duplicating the respective alternative and increasing the access time from the parking space to the desired destination by three minutes per duplicate, starting with duplicating the PUP alternative, and then the PSP alternative. For example: if we consider five offered alternatives, three of those will be PUP and two will be PSP. This extension corresponds to augmenting the parking space facilities in size and offering separate prices depending on proximity to the desired destination.

5.2 Description of experiments

The methods used in our experiments and their abbreviations are: Mixed-integer linear programming (MILP), Branch and Benders Decomposition (B&BD), B&BD without Benders decomposition (B&B), Breakpoint Exact Algorithm (BEA), Breakpoint Heuristic Algorithm (BHA), Iterated Local Search (ILS), the state-of-the-art heuristic for the capacitated CPP, the Lagrangian decomposition approach presented in Paneque et al. (2022), which we will refer to as LAG, and finally the state-of-the-art exact method for ML-based pricing without capacities, the convexification of a biconvex optimization and trust-region algorithm (CoBiT), presented by Marandi & Lurkin (2023). The goal of our experiments is to answer the following set of questions:

1. How does the BEA, adapted to capacity constraints, compare in runtime to the state-of-the-art exact MILP approach of solving capacitated instances of the CPP when an exogenous priority queue is set in place?

2. How do the BHA and ILS compare to the MILP and BEA approaches on instances with capacity constraints and a priority queue in terms of runtime and achieved revenue?
3. What are the largest instances we can solve within 72 hours using the BHA with capacity constraints?
4. How do the BHA and ILS compare to the state-of-the-art exact B&BD and BEA approaches on pricing instances without capacity constraints?
5. How does the number of alternatives impact the optimality gap for the BHA?
6. What are the largest instances we can solve within 72 hours using the BHA without capacity constraints?
7. To what extent are we able to speed up the B&BD method using the solution from the BHA for guidance, together with the derived valid inequalities?
8. How do the proposed solution methods compare to the LAG and CoBiT algorithms?

To investigate these eight issues we perform the tests described in Table 0, where N denotes the number of individuals considered, R the number of scenarios generated and J the number of controlled alternatives.

The limits for capacities are adapted from Paneque et al. (2021), as are the bounds for all prices, which are defined to be $[0.5, 0.7]$ for PSP alternatives and $[0.65, 0.85]$ for PUP alternatives. The initial starting point for the BHA in all cases is the mean of the bounds, i.e. $p_i^{\text{start}} = \frac{p_i^l + p_i^u}{2} \forall i \in C^1$. For the ILS, we use the following hyperparameter inputs: $\delta = 0.005$, $k = 3$, $\gamma = 2$, $\Delta_{\max} = 0.05$. Both the MILP and B&BD experiments are performed using GUROBI 10.0.3 (Gurobi Optimization, LLC, 2021). All methods are run on a single thread in a computational cluster node with two 2.4 GHz Intel Xeon Platinum 8360Y processors, where we utilize 16 cores with a total of 32 GB of RAM.

5.3 Numerical results and analysis

Table 1 illustrates results from Test 1, showing that the BEA with capacities is on average 20x faster than the MILP with equal revenues for all completed instances. For unsolved instances ($R = 250$), the BEA offers slightly better solutions. Comparing the BHA and ILS to the exact methods in Table 2, we find that the BHA is up to 360x faster than BEA for two prices, and ILS is up to 10x faster. For four prices, the BHA and ILS achieve speed-ups to 5000x and 65x, respectively. The ILS consistently finds global optima where verifiable but is about 40x slower than BHA, whose solutions are on average within less than 0.2% of the optimal value. For $R \geq 50$, the BHA matches the ILS's solutions, both finding higher quality solutions than the best feasible solution found by the exact methods at the time limit. Testing the limits of the BHA for capacitated instances, the results in Table 3 indicate that we can solve instances with up to 1,000 draws for 6 prices, up to

2,000 draws for 5 prices, and up to 3,000 prices for 4 prices or fewer within the time limit of 72 hours, marking a significant leap in instance size compared to the maximal number of 200 draws for two offered alternatives in Paneque et al. (2022).

For the uncapacitated case, Haering et al. (2023) have shown that instances with two or fewer prices can easily be solved with the BEA algorithm, whereas for instances with at least three alternatives, the B&BD approach is the fastest method. We thus consider only instances with at least three offered alternatives. Table 4 shows the results from Test 4, revealing that the BHA and ILS significantly outperform B&BD and BEA in terms of speed, with speedups of factor up to $3 \cdot 10^6$. The ILS, again about 40x slower than the BHA, does not improve objective values here, moreover, the BHA and ILS achieve the global optimum for all verifiable instances ($R = 100, 200$) and outperform the exact methods in terms of best solution found within the time limit in all unterminated cases. Table 5 depicts the outcomes of the analysis of the behavior of the optimality gap when the number of offered alternatives is increased. The instance size is kept small in order to be able to compute the global optimum with the B&BD algorithm within a reasonable time. We observe that generally, the mean of the bounds as a starting point does not lead to convergence to the global optimum, especially if the number of dimensions increases. However, the optimality gap is continuously very small and never reaches values bigger than 0.14%. We conclude that the BHA manages to deliver very high-quality solutions even for high-dimensional instances. The next test, whose results are found in Table 6, aims to understand the limits in terms of instance size for the BHA when no capacity constraints are set in place. Although a 72 hour time limit was set in place, the largest instance, $N = 50, R = 1,000,000, J = 6$ was solved in less than seven minutes, demonstrating that the BHA is capable of tackling much larger instances than the ones available from our current dataset, which should be addressed in future research. The outcomes of Test 7 are displayed over the next three tables, Tables 7, 8 and 9 respectively. They each depict the impact on computational time coming from adding the valid inequalities (VIs) to the McCormick relaxation in each node, the BHA solution as a starting point for the B&B search and the two enhancements combined, for achieving different optimality gaps. Table 7 shows that to reach an optimality gap of 10%, the total speedup from the enhancements over the unmodified B&BD reaches up to 50%, especially for instances with higher dimensions. Going further to 5% optimality in Table 8, the speedup reduces slightly and varies between 25-40%, decreasing further when looking at the computational time needed to reach 1% optimality gap in Table 9, which is reduced by 10-20% when using the enhancements. These speedups, where observable, remained stable until full convergence to the optimal solution (not depicted as only a fraction of these large instances were solved to optimality within the time limit). Two tendencies are clearly shown: For three alternatives, the VIs in fact seem to slow down the computation, however, increasing the number of alternatives shows that the more alternatives are being controlled, the larger the observed increase in speed when adding both the VIs or the heuristic starting point.

Regarding Test 8, we first compare our methods with the state-of-the-art heuristic method for general capacitated choice-based pricing, the LAG algorithm by Paneque et al. (2022). Unfortunately, the authors were not able to provide us with their code, which is why we

replicate their experimental environment (12 threads on a 3.33 GHz Intel Xeon X5680 server running a 64-bit Ubuntu 16.04.2) in order to be able to compare with the runtimes they provide in their paper. In their experiments, they run tests on the same parking space operator case study (with capacity constraints) in the following way: a time limit of two hours is set and the achieved optimality gap after the time limit is reported, but no objective values or optimal prices. This makes the comparison to our approaches difficult, as neither the exact BEA nor the heuristic BHA and ILS procedures report an optimality gap. Table 10 shows the comparison between the LAG, the BEA, the BHA and the ILS in terms of runtimes. The time limit is set to two hours and the instance sizes and capacity limitations are chosen in accordance with the tests conducted in Paneque et al. (2022). The LAG does not solve any of the instances to optimality within the time limit, the BEA manages to solve the smallest one ($N = 50, R = 50, J = 2$), the BHA converges for all instances and the ILS does so for four out of the nine tested instances. The BHA on average is at least 78x faster than the LAG, with the ILS on average being at least 4x faster. These factors are likely many times higher in reality, since closing the last few percentages of the optimality gap notoriously takes very long, which was also reported to be the case by Paneque et al. (2022). Comparing the numerical results is difficult, since for the LAG, we only have the achieved optimality gap. Table 11 thus shows the reported optimality gap for the LAG, the achieved revenues for the BEA, BHA and ILS, as well as the relative gap between the revenues of the BEA and BHA compared to the ILS, as we cannot report the real optimality gap. We note that within this time limit, both heuristic procedures produce better solutions than the exact BEA approach, with the average relative gap between BEA and ILS being 1.63%, and the average relative gap between the BHA and ILS being 0.04%. This further demonstrates that, although the BHA is significantly more efficient in terms of computational time, there is only a small loss in terms of solution quality when compared to the ILS algorithm. The average reported optimality gap of the LAG is with 3.19% pronouncedly larger than the average relative gap from BHA to ILS, however, the comparison should be contextualized carefully, considering the lack of an optimality certificate for the ILS.

We next compare to the state-of-the-art algorithm for (uncapacitated) mixed-logit-basing pricing, the CoBiT algorithm presented by Marandi & Lurkin (2023). The authors use the same parking choice case study as we do to illustrate their algorithm, and they have made their code available on GitHub. In their experiments, they address cases with $N = 10$ customer classes and two controlled prices, achieving optimal solutions. They approximate the continuous distributions of price sensitivity and arrival times with discrete distributions and report the runtimes based on varying numbers of breakpoints in their approximations.

Differing from our approach, their model assigns each pair of customer class and breakpoint to a customer under the MNL model, leading to nonlinear choice probabilities with respect to prices. Our method simulates each customer with deterministic preferences based on scenario draws. As a result, the problem we address for a given (N, R) differs and may not approximate the original model in the same way as their model with identical N and $R = n^2$ (number of breakpoints). According to numerical experiments

conducted in Haering et al. (2023), $R = 400$ draws in sample average simulation achieve similar accuracy to $n^2 = 64$ in Marandi & Lurkin (2023). Therefore, we consider good approximations with $n^2 = 64$ or $R = 400$, and smaller instances with $n^2 = 9$ or $R = 100$. We employ the same 10 customer classes as used in Marandi & Lurkin (2023), utilizing Julia version 1.8.0 and a computational cluster node with identical specifications to our other experiments, with all price bounds set to the closed unit interval $[0, 1]$ as per the authors' specifications, as well as no limitations on capacity. To ensure accurate runtime comparisons, we rerun the authors' method and successfully replicate their results. Additionally, we increase the number of controlled alternatives to up to six, as described in our previous experiments, to assess the impact of a larger choice set. The runtime results are presented in Table 12, and it is evident that the BHA is remarkably faster than CoBiT, by an average factor of $5.8 \cdot 10^6$, with an average optimality gap of 0.02%, whereas the ILS is faster by an average factor of 57,000, with an average optimality gap of 0.00%. Numerical results are detailed in the Appendix in Tables C1, C2, C3 and C4. It is important to mention a difference in how the price sensitivity parameters are generated between their framework and ours: in order to ensure behavioral realism, we opt to draw the factors β_p^{in} multiplying the prices using a truncated normal distribution, to guarantee that they are always negative. This additional step in the simulation procedure was not performed by Marandi & Lurkin (2023). Furthermore, the size of the confidence set for evaluating the integral (they chose a 99% set) influences the numerical value of the obtained revenue. It is for these reasons that there is a discrepancy in the optimal values for the prices and the objective between our approaches and CoBiT.

Table 0 – Summary of Tests

	Test 1	Test 2	Test 3	Test 4
J	2	2, 4	2, 4	2, 3, 4, 5, 6
N	50	50	50	50
R	2, 5, 10, 25, 50, 100, 250	2, 5, 10, 25, 50, 100, 200, 250	2, 5, 10, 25, 50, 100, 200, 250	1,000, 2,000, 3000
Capacities	[20, 20]	[20, 20], [15, 15, 15, 15]	[20, 20], [15, 15, 15, 15]	[20] $\forall i \in C_n$
Methods	MILP, BEA	BEA, BEA-M, BEA-R	MILP, BEA, BHA, ILS	BHA

	Test 5	Test 6	Test 7	Test 8
J	4	3, 4, 5, 6, 7, 8, 9, 10	2, 3, 4, 5, 6	2, 3, 4, 5, 6
N	20	20	50	10, 50, 100, 150, 197
R	100, 200, 300, 500, 1,000	20	500,000, 1,000,000	25, 50, 100, 200, 400
Capacities	[∞ , ∞ , ∞ , ∞]	[∞] $\forall i \in C_n$	[∞] $\forall i \in C_n$	[20], [40], [60], [80], [∞] $\forall i \in C_n$
Methods	B&BD, BEA, BHA, ILS	B&BD, BHA	BHA	LAG, CoBiT, BEA, BEA, BHA, ILS, B&B, B&BD

Table 1 – Test 1: MILP vs. BEA in the capacitated case

N	R	J	MILP		BEA	
			Time (s)	Revenue	Time (s)	Revenue
50	2	2	4.17	27.61	0.43	27.61
50	5	2	46.95	26.51	1.72	26.51
50	10	2	180.85	27.06	11.42	27.06
50	25	2	3,119.66	27.08	169.08	27.08
50	50	2	>5 hours	≥ 25.15	1,272.68	26.85
50	100	2	>25 hours	≥ 25.11	9,928.57	26.85
50	250	2	>45 hours	≥ 23.45	>45 hours	≥ 26.37

Table 2 – Test 2: BHA and ILS vs. MILP and BEA in the capacitated case

N	R	J	MILP		BEA		BHA		ILS	
			Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue
50	2	2	4.17	27.61	0.43	27.61	0.22	27.61	1.03	27.61
50	5	2	46.95	26.51	1.72	26.51	0.32	26.46	5.91	26.51
50	10	2	180.85	27.06	11.42	27.06	0.58	27.05	20.34	27.06
50	25	2	3,119.66	27.08	169.08	27.08	3.40	27.05	129.66	27.08
50	50	2	>5 hours	≥ 25.15	1,272.68	26.85	8.31	26.53	559.04	26.85
50	100	2	>25 hours	≥ 25.11	9,928.57	26.85	51.77	26.72	2,791.28	26.85
50	250	2	>45 hours	≥ 23.45	>45 hours	≥ 26.37	455.37	26.66	15,867.67	26.71
50	10	4	>10 hours	≥ 22.21	>10 hours	≥ 25.10	7.08	26.78	527.34	26.83
50	50	4	>20 hours	≥ 22.19	>20 hours	≥ 25.19	166.21	27.00	7,234.88	27.00
50	100	4	>45 hours	≥ 20.50	>45 hours	≥ 26.09	866.97	26.67	34,050.57	26.67
50	200	4	>72 hours	≥ 20.32	>72 hours	≥ 24.79	2,762.39	26.70	106,286.13	26.70

Table 3 – Test 3: Limits of the BHA in the capacitated case

N	R	J	BHA (s)
50	1,000	2	15,093
50	1,000	3	25,326
50	1,000	4	69,134
50	1,000	5	112,042
50	1,000	6	178,923
50	2,000	2	51,637
50	2,000	3	84,231
50	2,000	4	150,132
50	2,000	5	193,233
50	3,000	2	164,922
50	3,000	3	184,293
50	3,000	4	>259,200

Table 4 – Test 4: BHA and ILS vs. B&BD and BEA in the uncapacitated case

N	R	J	B&BD		BEA		BHA		ILS	
			Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue	Time (s)	Revenue
20	100	4	12,478	10.14	61,139	10.14	0.00	10.14	0.14	10.14
20	200	4	29,213	10.40	>24 hours	≥ 10.21	0.01	10.40	0.41	10.40
20	300	4	>24 hours	≥ 10.38	>24 hours	≥ 9.84	0.02	10.24	0.64	10.24
20	400	4	>24 hours	≥ 9.81	>24 hours	≥ 9.82	0.05	10.26	0.78	10.26
20	500	4	>24 hours	≥ 10.01	>24 hours	≥ 9.80	0.13	10.24	1.37	10.24

Table 5 – Test 5: BHA optimality gap when increasing dimensions

N	R	J	BHA	B&BD	Gap (%)
20	20	3	10.281	10.281	0
20	20	4	10.271	10.28	0.09
20	20	5	10.283	10.294	0.11
20	20	6	10.290	10.302	0.12
20	20	7	10.292	10.306	0.14
20	20	8	10.330	10.336	0.06
20	20	9	10.329	10.335	0.06
20	20	10	10.293	10.300	0.07

Table 6 – Test 6: Testing BHA limits without capacity constraints

N	R	J	BHA (s)
50	500,000	2	56
50	500,000	3	77
50	500,000	4	187
50	500,000	5	163
50	500,000	6	194
50	1,000,000	2	68
50	1,000,000	3	132
50	1,000,000	4	312
50	1,000,000	5	300
50	1,000,000	6	412

Table 7 – Test 7: B&BD with Guidance - 10% gap

N	R	J	normal w/out VIs (s)	normal w VIs (s)	Guided w/out VIs (s)	Guided w VIs (s)	Speedup from just VIs (%)	Add. Speedup from Sol. (%)	Total speedup (%)
50	1,000	3	987	1,132	731	816	-14.69	27.92	17.33
50	2,000	3	2,878	3,490	2,513	2,693	-21.26	22.84	6.43
50	3,500	3	10,325	12,919	6,390	7,454	-25.12	42.3	27.81
50	1,000	4	4,662	3,311	3,705	2,472	28.98	25.34	46.98
50	2,000	4	17,599	12,068	10,868	8,288	31.43	31.32	52.91
50	3,500	4	48,445	31,210	40,061	29,504	35.58	5.47	39.1
50	1,000	5	8,242	5,428	5,664	3,914	34.14	27.89	52.51
50	2,000	5	25,842	16,641	17,420	12,268	35.6	26.28	52.53
50	3,500	5	114,216	81,826	85,083	58,754	28.36	28.2	48.56

Table 8 – Test 7: B&BD with Guidance - 5% gap

N	R	J	normal w/out VIs (s)	normal w VIs (s)	Guided w/out VIs (s)	Guided w VIs (s)	Speedup from just VIs (%)	Add. Speedup from Sol. (%)	Total speedup (%)
50	1,000	3	2,372	2,454	1,933	2,245	-3.46	8.52	5.35
50	2,000	3	7,883	8,359	7,106	7,342	-6.04	12.17	6.86
50	3,500	3	51,964	57,229	42,991	47,282	-10.13	17.38	9.01
50	1,000	4	12,062	10,668	10,490	8,934	11.56	16.25	25.93
50	2,000	4	43,829	36,524	36,222	32,929	16.67	9.84	24.87
50	3,500	4	259,200	240,767	238,777	198,981	7.11	17.36	23.23
50	1,000	5	24,371	20,590	19,519	16,930	15.51	17.78	30.53
50	2,000	5	84,104	60,814	70,676	48,541	27.69	20.18	42.28
50	3,500	5	259,200	259,200	259,200	247,944	-	-	-

Table 9 – Test 7: B&BD with Guidance - 1% gap

N	R	J	normal w/out VIs (s)	normal w VIs (s)	Guided w/out VIs (s)	Guided w VIs (s)	Speedup from just VIs (%)	Add. Speedup from Sol. (%)	Total speedup (%)
50	1,000	3	15,840	16,933	13,239	14,594	-6.9	13.81	7.87
50	2,000	3	42,261	45,223	35,882	37,137	-7.01	17.88	12.12
50	3,500	3	183,696	195,743	152,833	162,594	-6.56	16.93	11.49
50	500	4	47,101	46,719	47,963	43,190	0.81	7.55	8.3
50	1,000	4	131,122	135,564	107,288	105,596	-3.39	22.11	19.47
50	1,500	4	229,620	230,187	203,348	202,560	-0.25	12	11.78
50	2,000	4	259,200	259,200	259,200	259,200	-	-	-
50	500	5	139,618	125,755	115,783	109,084	9.93	13.26	21.87
50	1,000	5	259,200	259,200	259,200	259,200	-	-	-

Table 10 – Test 8: Runtime (seconds) comparison to LAG

N	R	J	LAG	BEA	BHA	x Sp.	ILS	x Sp.
50	50	2	>7,200	3,109	24	300	841	9
50	100	2	>7,200	>7,200	96	75	3,640	2
50	200	2	>7,200	>7,200	459	16	>7,200	-
100	100	2	>7,200	>7,200	554	13	>7,200	-
150	100	2	>7,200	>7,200	1166	6	>7,200	-
197	100	2	>7,200	>7,200	1617	4	>7,200	-
50	25	4	>7,200	>7,200	31	230	2,711	3
50	50	4	>7,200	>7,200	148	49	7,157	1
50	100	4	>7,200	>7,200	591	12	>7,200	-

Table 11 – Test 8: Objective value comparison to LAG (with a two-hour time limit)

N	R	J	LAG	BEA		BHA		ILS
			Gap (%)	Revenue	Gap* (%)	Revenue	Gap* (%)	Revenue
50	50	2	2.02	26.243	0.00	26.237	0.02	26.243
50	100	2	2.80	26.560	1.28	26.906	0.00	26.906
50	200	2	3.67	26.250	1.14	26.530	0.09	26.553
100	100	2	1.98	52.780	0.53	53.028	0.06	53.059
150	100	2	1.91	80.370	0.33	80.640	0.00	80.640
197	100	2	-	104.640	0.70	105.181	0.19	105.381
50	25	4	3.34	25.728	4.26	26.873	0.00	26.873
50	50	4	4.57	25.180	3.52	26.099	0.00	26.099
50	100	4	5.19	26.090	2.90	26.870	0.00	26.870

* relative gap compared to ILS revenue

Table 12 – Test 8: Runtime (seconds) comparison to CoBiT

N	n ²	R	J	CoBiT	B&B	x Sp.	B&BD	x Sp.	BEA	x Sp.	BHA	x Sp.	ILS	x Sp.
10	9	100	2	69	17	4	83	0.83	1	69	0.002	4 · 10 ⁴	0.086	971
10	9	100	3	607	124	5	623	0.97	10	61	0.001	4.5 · 10 ⁵	0.117	5,343
10	9	100	4	6,439	985	7	4,791	1.34	5727	1	0.002	3.2 · 10 ⁶	0.216	2.2 · 10 ⁴
10	9	100	5	34,409	4,017	9	18,644	1.85	>86,400	-	0.001	27 · 10 ⁶	0.250	7.4 · 10 ⁴
10	9	100	6	39,164	6,015	7	27,758	1.41	>86,400	-	0.002	15 · 10 ⁶	0.254	1.1 · 10 ⁵
10	64	400	2	270	128	2	620	0.44	1	270	0.003	8.6 · 10 ⁴	0.139	4,461
10	64	400	3	4,234	783	5	3,174	1.33	560	8	0.005	9.2 · 10 ⁵	0.244	1.2 · 10 ⁴
10	64	400	4	37,384	8,895	4	34,503	1.08	>86,400	-	0.010	3.8 · 10 ⁶	0.460	7.5 · 10 ⁴
10	64	400	5	38,090	25,367	2	>86,400	-	>86,400	-	0.010	3.6 · 10 ⁶	0.658	1.3 · 10 ⁵
10	64	400	6	39,424	>86,400	-	>86,400	-	>86,400	-	0.012	3.4 · 10 ⁶	0.650	1.3 · 10 ⁵

5.4 Summary of results

We summarize our findings as follows: The BEA, adapted to capacity constraints, significantly outperforms the MILP formulation for capacitated instances of the CPP, being up to 20 times faster while delivering equivalent revenues. Even when the MILP fails ($R = 250$), BEA still offers slightly better solutions. The BHA and ILS heuristics show remarkable speed improvements, with BHA being up to 360 times faster than BEA for two prices and 5000 times faster for four prices. Although ILS is slower, it consistently achieves global optima, while BHA stays within 0.2% of optimal values. Both outperform exact methods for $R \geq 50$.

The BHA handles large capacitated instances effectively, solving problems with up to 1,000 draws for 6 prices within 72 hours, far surpassing previous methods. In uncapacitated scenarios, BHA and ILS outshine exact methods like B&BD and BEA, achieving speedups up to $3 \cdot 10^6$ while consistently finding global optima. BHA maintains a minimal optimality gap (never exceeding 0.14%), even for high-dimensional instances, and solves very large instances (one million draws) in under seven minutes. Enhancements to the B&BD method, using BHA solutions and valid inequalities, provide speedups of up to 50% depending on the optimality gap, but are less effective as the gap tightens.

Compared to state-of-the-art methods, the BHA offers significant improvements, outperforming the LAG algorithm (with capacities) by a factor of at least 78 in runtime and CoBiT (without capacities) by $5.8 \cdot 10^6$ in mixed-logit scenarios, with an average optimality gap of less than 0.02%. The ILS delivers remarkable speedups as well, while maintaining optimality. Overall, the BHA exhibits exceptional performance across instances both with and without capacity constraints and effectively addresses problems in both low and high-dimensional settings.

6 Conclusions

This research introduces the Breakpoint Heuristic Algorithm (BHA), which offers a substantial advancement in solving the choice-based pricing problem (CPP) with and without capacity constraints, as well as valid inequalities to tighten the relaxation in the state-of-the-art approach for the uncapacitated problem, the Branch and Benders Decomposition (B&BD) approach. To handle capacities, we first extend the Breakpoint Exact Algorithm (BEA) with a capacity management strategy, implemented using an exogenous priority queue. This adapted algorithm outperforms the state-of-the-art mixed-integer linear programming (MILP) approach by 20 times in computational speed for simultaneous two-price optimization. The BHA, employing a coordinate descent method, excels in high-dimensional scenarios, showing remarkable efficiency in both capacitated and uncapacitated cases. Notably, it outpaces the MILP by a factor of up to 5000 for the capacitated case, and the B&BD approach by several orders of magnitude for the uncapacitated case, while maintaining an average optimality gap of less than 0.2%. It can furthermore be flexibly adapted to various use cases, due to its independent objective value evaluation. The iterated local search (ILS) extension of the BHA succeeds in identifying the global

optimum in all tested instances, albeit with a significant speed reduction (average factor of 40x). Combining the new valid inequalities with the solution found by the BHA to guide the B&BD algorithm leads to decreases in computational time of up to 50%, with high-dimensional instances showing the largest improvements. The BHA significantly surpasses state-of-the-art methods, including those tailored for mixed logit models, in speed while maintaining an optimality gap of less than 0.02%.

We thus conclude that we have successfully contributed to filling the gaps identified in the literature: we provide operational algorithms to solve the CPP, capable of handling large instance sizes and complex additional constraints (like capacity), while maintaining only weak assumptions, if any, on the choice model.

Various avenues are open for future research: In terms of the heuristic, other extensions of the BHA to escape local optima should be considered, as the ILS increases computational time substantially. The remarkable speed of the BHA algorithm, together with its capability to produce high-quality solutions and overall flexibility due to depending only on evaluating an objective function given fixed parameters, lays the groundwork for its application in larger as well as more intricate problem settings.

References

- Abdolhamidi, D., & Lurkin, V. (2024). A tactical time slot management problem under mixed logit demand. *arXiv preprint arXiv:2407.02308*.
- Benati, S., & Hansen, P. (2002). The maximum capture problem with random utilities: Problem formulation and algorithms. *European Journal of operational research*, 143(3), 518–530.
- Binder, S., Maknoon, Y., & Bierlaire, M. (2017). Exogenous priority rules for the capacitated passenger assignment problem. *Transportation Research Part B: Methodological*, 105, 19–42. (Accepted on Aug 17, 2017) doi: 10.1016/j.trb.2017.08.022
- Bortolomiol, S., Lurkin, V., & Bierlaire, M. (2021). A simulation-based heuristic to find approximate equilibria with disaggregate demand models. *Transportation Science*, 55(5), 1025–1045.
- Cordone, R., & Redaelli, F. (2011). Optimizing the demand captured by a railway system with a regular timetable. *Transportation Research Part B: Methodological*, 45(2), 430–446.
- Davis, J. M., Topaloglu, H., & Williamson, D. P. (2017). Pricing problems under the nested logit model with a quality consistency constraint. *INFORMS Journal on Computing*, 29(1), 54–76.
- Gallego, G., & Wang, R. (2014). Multiproduct price optimization and competition under the nested logit model with product-differentiated price sensitivities. *Operations Research*, 62(2), 450–461.

- Gilbert, F., Marcotte, P., & Savard, G. (2014). Mixed-logit network pricing. *Computational Optimization and Applications*, 57, 105–127.
- Gosavi, A. (2015). *Simulation-based optimization* (Vol. 55). doi: 10.1007/978-1-4899-7491-4
- Gurobi Optimization, LLC. (2021). *Gurobi Optimizer Reference Manual*. Retrieved from <https://www.gurobi.com>
- Haase, K., & Müller, S. (2013). Management of school locations allowing for free school choice. *Omega*, 41(5), 847–855.
- Haering, T., Legault, R., Torres, F. A., Ljubic, I., & Bierlaire, M. (2023). Exact algorithms for continuous pricing with advanced discrete choice demand models. *OR Spectrum*. (Accepted for publication)
- Hanson, W., & Martin, K. (1996). Optimizing multinomial logit profit functions. *Management Science*, 42(7), 992–1003.
- Ibeas, A., Dell’Olio, L., Bordagaray, M., & Ortúzar, J. d. D. (2014). Modelling parking choices considering user heterogeneity. *Transportation Research Part A: Policy and Practice*, 70, 41–49.
- Korfmann, F. (2018). *Essays on advanced discrete choice applications* (Unpublished doctoral dissertation). Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky.
- Legault, R., & Frejinger, E. (2024). A model-free approach for solving choice-based competitive facility location problems using simulation and submodularity. *INFORMS Journal on Computing*.
- Li, H., Webster, S., Mason, N., & Kempf, K. (2019). Product-line pricing under discrete mixed multinomial logit demand: winner—2017 msom practice-based research competition. *Manufacturing & Service Operations Management*, 21(1), 14–28.
- Liberti, L. (2008). Introduction to global optimization. *Ecole Polytechnique*.
- Liu, N., Ma, Y., & Topaloglu, H. (2020). Assortment optimization under the multinomial logit model with sequential offerings. *INFORMS Journal on Computing*, 32(3), 835–853.
- Ljubić, I., & Moreno, E. (2018). Outer approximation and submodular cuts for maximum capture facility location problems with random utilities. *European Journal of Operational Research*, 266(1), 46–56.
- Mai, T., & Lodi, A. (2020). A multicut outer-approximation approach for competitive facility location under random utilities. *European Journal of Operational Research*, 284(3), 874–881.

- Marandi, A., & Lurkin, V. (2023). An exact algorithm for the static pricing problem under discrete mixed logit demand. *EURO Journal on Computational Optimization*, *11*, 100073.
- McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, *10*(1), 147–175.
- Müller, D., Nesterov, Y., & Shikhman, V. (2021). Dynamic pricing under nested logit demand. *arXiv preprint arXiv:2101.04486*.
- Paneque, M. P., Bierlaire, M., Gendron, B., & Azadeh, S. S. (2021). Integrating advanced discrete choice models in mixed integer linear optimization. *Transportation Research Part B: Methodological*, *146*, 26–49.
- Paneque, M. P., Gendron, B., Azadeh, S. S., & Bierlaire, M. (2022). A lagrangian decomposition scheme for choice-based optimization. *Computers & Operations Research*, *148*, 105985.
- Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*(3), 801–817.
- Robenek, T., Azadeh, S. S., Maknoon, Y., de Lapparent, M., & Bierlaire, M. (2018). Train timetable design under elastic passenger demand. *Transportation research Part b: methodological*, *111*, 19–38.
- Rusmevichientong, P., Shen, Z.-J. M., & Shmoys, D. B. (2010). Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research*, *58*(6), 1666–1680.
- Shen, Z.-J. M., & Su, X. (2007). Customer behavior modeling in revenue management and auctions: A review and new research opportunities. *Production and operations management*, *16*(6), 713–728.
- Sumida, M., Gallego, G., Rusmevichientong, P., Topaloglu, H., & Davis, J. (2021). Revenue-utility tradeoff in assortment optimization under the multinomial logit model with totally unimodular constraints. *Management Science*, *67*(5), 2845–2869.
- van de Geer, R., & den Boer, A. V. (2022). Price optimization under the finite-mixture logit model. *Management Science*, *68*(10), 7480–7496.
- Wu, D., Yin, Y., Lawphongpanich, S., & Yang, H. (2012). Design of more equitable congestion pricing and tradable credit schemes for multimodal transportation networks. *Transportation Research Part B: Methodological*, *46*(9), 1273–1287.

Appendix A Extension to capacity constraints

Formulation 1 can be extended to incorporate capacity constraints on the alternatives, as demonstrated in Paneque et al. (2021). However, adding these constraints makes it no longer possible to relax the domain of the ω variables, meaning the problem has to be written as a MILP. Furthermore, some constraints presented in Formulation 1 have to be adjusted. For the sake of clarity, we present the full MILP formulation for the CPP with capacity constraints, as it represents the state-of-the-art to model and solve the problem.

First, we assume a set of given capacities $c_i, i \in C^1$. We now introduce two new sets of variables, the binary y_{inr} variables, which indicate the availability of alternative $i \in C^1$ for individual n in scenario r , and the discounted utility variables z_{inr} , which are equal to the utilities U_{inr} if alternative $i \in C^1$ is available (i.e. $y_{inr} = 1$) and set to a low enough value otherwise, thus embedding the concept of unavailable alternatives in the customer subproblem. We note that individuals are assumed to be numbered in the order of their priority to access the market. This numbering can follow any rule, including being random, but must be exogenous Binder et al. (2017). Lastly, as the choice variables are now again binary, the product $p_i \omega_{inr}$ can be modeled in a linear way using big-M constraints, where the optimal big M is the largest possible value taken by said product, i.e. p_i^U . The full MILP formulation for the capacity-constrained CPP is given in Formulation 3:

$$\begin{aligned}
& \max_{p, \omega, \eta, \mathbf{U}, \mathbf{h}} \frac{1}{R} \sum_{r \in \mathcal{R}} \sum_{n \in \mathcal{N}} \sum_{i \in C_n^1} \eta_{inr} \\
& \text{s.t.} \\
& \sum_{i \in C_n} \omega_{inr} = 1 \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\mu_{nr}) \\
& h_{nr} \leq z_{inr} + M(1 - \omega_{inr}) \quad \forall n \in \mathcal{N}, i \in C_n, r \in \mathcal{R} \quad (\zeta_{nr}) \\
& h_{nr} \geq z_{inr} \quad \forall n \in \mathcal{N}, i \in C_n, r \in \mathcal{R} \quad (\alpha_{inr}) \\
& \mathbf{U}_{inr} = \mathbf{c}_{inr} + \beta_p^{\text{in}} \mathbf{p}_i \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\kappa_{inr}) \\
& \eta_{inr} \leq \omega_{inr} \mathbf{p}_i^{\text{U}} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^1) \\
& \eta_{inr} \leq \mathbf{p}_i \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^2) \\
& \eta_{inr} \geq \mathbf{p}_i - (1 - \omega_{inr}) \mathbf{p}_i^{\text{U}} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\lambda_{inr}^3) \\
& z_{0nr} = \mathbf{c}_{0nr} \quad \forall n \in \mathcal{N}, r \in \mathcal{R} \quad (\mathbf{a}_{nr}) \\
& z_{inr} \leq \mathbf{U}_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\mathbf{b}_{inr}) \\
& z_{inr} \geq \mathbf{U}_{inr} - M(1 - \mathbf{y}_{inr}) \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\mathbf{c}_{inr}) \\
& z_{inr} \leq M/2 + M\mathbf{y}_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\mathbf{d}_{inr}) \\
& \omega_{inr} \leq \mathbf{y}_{inr} \quad \forall n \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\mathbf{e}_{inr}) \\
& \sum_{m=1}^n \omega_{imr} \leq (\mathbf{c}_i - 1)\mathbf{y}_{inr} + (n-1)(1 - \mathbf{y}_{inr}) \quad \forall n > \mathbf{c}_i \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\mathbf{f}_{inr}) \\
& \sum_{m=1}^n \omega_{imr} \geq \mathbf{c}_i(1 - \mathbf{y}_{inr}) \quad \forall n > 1 \in \mathcal{N}, i \in C_n^1, r \in \mathcal{R} \quad (\mathbf{g}_{inr}) \\
& \omega \in \{0, 1\}^{(J+1)NR} \\
& \mathbf{y} \in \{0, 1\}^{JNR} \\
& \mathbf{p} \in [p_1^{\text{L}}, p_1^{\text{U}}] \times \dots \times [p_J^{\text{L}}, p_J^{\text{U}}] \\
& \eta \in [0, p_1^{\text{U}}] \times \dots \times [0, p_J^{\text{U}}] \\
& \mathbf{U}, \mathbf{z}, \mathbf{h} \in \mathbb{R}^{JNR}, \mathbb{R}^{(J+1)NR}, \mathbb{R}^{NR}
\end{aligned} \tag{3}$$

where M is a large enough constant. Constraints (ζ_{nr}) and (α_{inr}) have been adjusted to incorporate the newly added z variables. The non-convex constraints (λ_{inr}) have been replaced by a set of linear constraints (λ_{inr}^1) , (λ_{inr}^2) and (λ_{inr}^3) , modeling the big-M linearization of the aforementioned product. Constraints $(\mathbf{a}_{inr} - \mathbf{d}_{inr})$ define the discounted utility z_{inr} , ensuring that in case of $\mathbf{y}_{inr} = 0$, z_{inr} takes on a small enough value to not compete with any available alternative's utility, and is set to be equal to \mathbf{U}_{inr} otherwise. Constraint (\mathbf{e}_{inr}) enforces that an alternative can only be chosen if it is available to that individual in that scenario. Lastly, constraints (\mathbf{f}_{inr}) and (\mathbf{g}_{inr}) define the \mathbf{y} variables: Constraint (\mathbf{f}_{inr}) sets \mathbf{y}_{inr} to 0 whenever the capacity of alternative i is reached and con-

straint (g_{inr}) makes sure that, if an alternative is no longer available, there have to be enough people choosing the alternative to fill it to its capacity.

Appendix B Extending the Breakpoint Exact Algorithm (BEA) to handle capacity constraints

To incorporate capacity constraints into the BEA, we employ a streamlined variant that systematically explores each valid combination of breakpoints in sequence. We use the same notation as Haering et al. (2023). As for the uncapacitated case, the optimal price ensures that the utility of a product matches the utility of the next cheapest alternative for at least one customer and scenario, maximizing revenue without unnecessary customer loss. Specifically, for an optimal price $p_i, i \in C^1$, there exists a customer $n \in N$ and scenario $r \in R$ such that any increase in p_i by $\varepsilon > 0$ would lower the utility U_{inr} below that of cheaper alternatives or the opt-out option, deterring that customer and decreasing overall revenue. Hence, this price acts as a “breakpoint” or “indifference point” in the customer’s decision-making, representing the maximum price before their interest shifts to more affordable options.

The outer level algorithm (see Algorithm 4) remains the same as for the BEA, with the only difference being that the recursive `enumerate` function is replaced by a function `enumerate_cap`, described in Algorithm 5. At its deepest level, it invokes the function `compute_objective_value` that calculates the objective value for a set price variables fixed to a combination of breakpoints, taking capacity restrictions into account. Due to interdependent choices potentially leading to recursive substitution, continuously updating choices and revenues becomes impractical. Additionally, when adding a new product, calculating breakpoints for each simulated customer from their previous preference to the new option is insufficient in this problem setting. Instead, breakpoints must be computed from any possible previous product to the new one, as capacity limits may force customers to choose an alternative other than their most preferred. This adjustment accounts for decision breakpoints involving switches from any introduced product to the new one. The process of sequentially introducing alternatives remains the same as in the original BEA, as the order in which alternatives are introduced to customers alters their decision making breakpoints. A diagram visualizing the BEA with capacity constraints is shown in Figure 1.

These two changes, compared to the BEA without capacity constraints, increase the algorithm’s computational complexity. However, directly evaluating the objective function at each breakpoint combination also enhances flexibility in revenue computation methods. Indeed, this allows to add any type of constraint to the problem without adjusting the algorithm. In our case, we only consider capacity constraints, implemented with an exogenous priority queue. Algorithm 3 lays out the evaluation of revenue given such a queue, where individuals are assigned the highest utility alternative with positive remaining capacity.

Algorithm 3: Compute objective value with priority queue

```
Function compute_objective_value (p, c, prio_queue) :  
   $\sigma \leftarrow (0)_{i \in C}$   
  for idx  $\in$  prio_queue do  
     $u \leftarrow [U_{idx}^i \text{ for } i \in C]$   
     $a \leftarrow \text{sort}(u, \text{descending})$   
     $\varphi \leftarrow \text{false}$   
     $j \leftarrow 1$   
    while  $j \leq C - 1$  and  $!\varphi$  do  
      if  $\sigma_{a_j} \leq c_{a_j} - 1$  then  
         $\sigma_{a_j} += 1$   
         $\varphi \leftarrow \text{true}$   
      end  
      else  
         $j += 1$   
      end  
    end  
  end  
   $o \leftarrow \sum_{i \in C} \sigma_i \cdot p_i$   
  return o  
end
```

Algorithm 4: Breakpoint exact algorithm (BEA) to solve the capacitated CPP

```
Result: optimal solution  $p^*$  and objective value  $o^*$  for the capacitated CPP.  
 $p_j^* \leftarrow 0 \quad \forall j \in \{1, \dots, J\}$   
 $o^* \leftarrow 0$   
for  $s$  in  $S$  do  
   $p_{s_j} \leftarrow 0 \quad \forall j \in \{1, \dots, J\}$   
   $(\hat{p}, \hat{o}) \leftarrow \text{enumerate\_cap}(s, p, 1)$   
  if  $\hat{o} > o^*$  then  
     $p^* \leftarrow \hat{p};$   
     $o^* \leftarrow \hat{o};$   
  end  
end  
return  $(p^*, o^*)$ 
```

We denote by S the set of all possible permutations s of $\{1, \dots, J\}$. For a given permutation $s \in S$, the j^{th} element of the ordered list s is denoted by s_j . Algorithm 4 iterates over all possible orderings of prices $p_{s_1} \leq p_{s_2} \leq \dots \leq p_{s_J}$, $s \in S$. Each restricted problem is addressed by the recursive `enumerate_cap` function. This function accepts as arguments the current permutation $s \in S$ of alternatives, a partially filled vector of prices p , with components $p_{s_1} \leq \dots \leq p_{s_{j-1}}$ already set, and the depth j of the current permutation's exploration.

The function `enumerate_cap` solves the capacitated CPP, restricted to a specific ordering of prices, in a recursive manner.

Algorithm 5: Recursive enumeration function within the BEA, when taking capacities into account

```

Function enumerate_cap (s, p, j) :
   $\bar{p}_{s_j}^{nrs_i} \leftarrow \frac{u_{nr}^{s_i} - c_{s_j nr}}{\beta_p^{s_j n}} \quad \forall (n, r) \in \mathcal{N} \times \mathcal{R}, i < j \in \mathcal{C} \cup \{0\}$ 
   $\mathcal{N}_2 \leftarrow \{(n, r, s_i) | p_{s_j}^L < \bar{p}_{s_j}^{nrs_i} < p_{s_j}^U\}$ 
   $\mathcal{N}_2 \leftarrow \mathcal{N}_2 \cup \{p_{s_j}^L, p_{s_j}^U \mid \forall i \in \mathcal{C}\}$ 
  Sort the elements of  $\mathcal{N}_2$  from largest to smallest
  if  $j \leq J - 1$  then
    for  $\bar{p}_{s_j}^{nrs_i} \in \mathcal{N}_2$  do
       $p_{s_j} \leftarrow \bar{p}_{s_j}^{nrs_i}$ 
       $(\hat{p}, \hat{o}) \leftarrow \text{enumerate\_cap}(s, p, j + 1)$ 
      if  $\hat{o} > o^*$  then
         $o^* \leftarrow \hat{o}$ 
         $p^* \leftarrow \hat{p}$ 
      end
    end
  end
  else
    for  $\bar{p}_{s_j}^{nrs_i} \in \mathcal{N}_2$  do
       $p_{s_j} \leftarrow \bar{p}_{s_j}^{nrs_i}$ 
       $o \leftarrow \text{compute\_objective\_value}(p)$ 
      if  $o > o^*$  then
         $o^* \leftarrow o$ 
         $p^* \leftarrow p$ 
      end
    end
  return  $(p^*, o^*)$ 
end
end

```

The exponential growth in BEA's complexity with the number of controlled alternatives J is clearly illustrated in the tree diagram in Figure 1, where each branch represents a call to the recursive `enumerate_cap` function.

Appendix C Comparison to CoBiT

This section shows the numerical results for the CoBiT, B&B, BHA and ILS algorithms when applied to the artificially augmented parking choice data set.

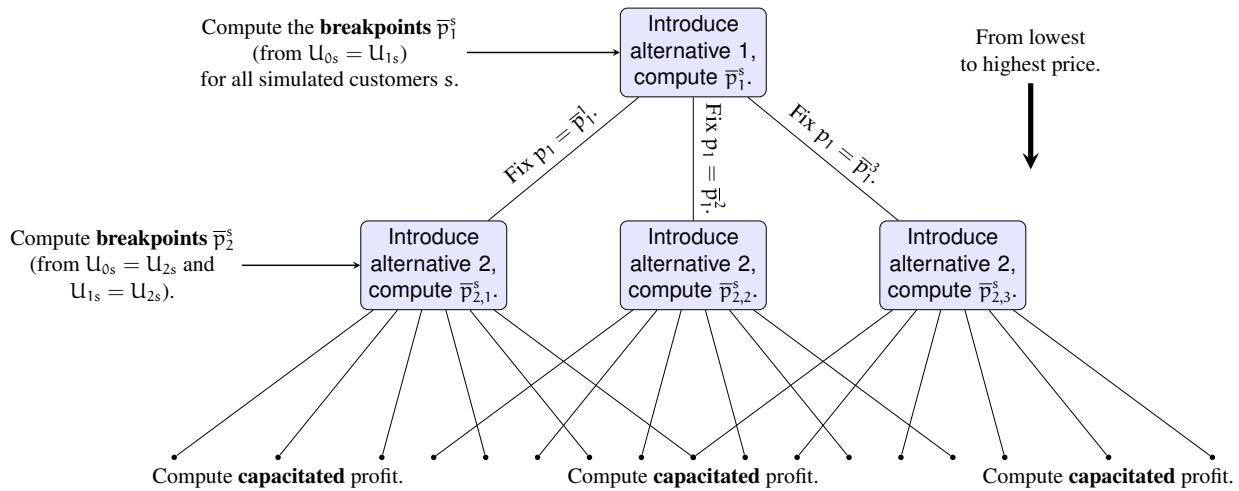


Figure 1 – BEA with capacity constraints for three simulated customers and two alternatives.

Table C1 – Numerical results for CoBiT

N	R	J	Obj.	p_1	p_2	p_3	p_4	p_5	p_6
10	9	2	6.837	0.520	0.729				
10	9	3	6.761	0.500	0.690	1.000			
10	9	4	6.882	0.531	0.719	0.750	0.520		
10	9	5	6.842	0.540	0.722	0.750	0.500	0.697	
10	9	6	6.839	0.540	0.720	0.750	0.500	0.950	0.500
10	64	2	5.069	0.500	0.661				
10	64	3	5.080	0.500	0.659	0.664			
10	64	4	5.086	0.500	0.659	0.664	0.500		
10	64	5	5.084	0.500	0.662	0.660	0.500	0.625	
10	64	6	5.086	0.500	0.661	0.661	0.500	0.628	0.498

Table C2 – Numerical results for best exact method (B&B)

N	R	J	Obj.	p ₁	p ₂	p ₃	p ₄	p ₅	p ₆
10	100	2	5.200	0.626	0.651				
10	100	3	5.142	0.562	0.652	0.674			
10	100	4	5.140	0.543	0.560	0.652	0.677		
10	100	5	5.142	0.543	0.560	0.652	0.677	0.668	
10	100	6	5.155	0.540	0.575	0.530	0.652	0.656	0.680
10	400	2	5.279	0.550	0.652				
10	400	3	5.196	0.550	0.652	0.651			
10	400	4	5.204	0.549	0.570	0.652	0.655		
10	400	5	5.208	0.549	0.570	0.652	0.655	0.650	
10	400	6	5.235	0.564	0.556	0.538	0.650	0.661	0.669

Table C3 – Numerical results for BHA

N	R	J	Obj.	Gap (%)	p ₁	p ₂	p ₃	p ₄	p ₅	p ₆
10	100	2	5.200	0.00	0.626	0.651				
10	100	3	5.141	0.02	0.548	0.653	0.673			
10	100	4	5.140	0.00	0.543	0.560	0.652	0.677		
10	100	5	5.142	0.00	0.540	0.561	0.651	0.673	0.665	
10	100	6	5.152	0.05	0.543	0.579	0.654	0.653	0.657	0.681
10	400	2	5.279	0.00	0.550	0.652				
10	400	3	5.196	0.00	0.550	0.653	0.651			
10	400	4	5.201	0.06	0.532	0.534	0.650	0.658		
10	400	5	5.205	0.07	0.532	0.534	0.650	0.658	0.650	
10	400	6	5.234	0.03	0.564	0.558	0.542	0.650	0.661	0.669

Table C4 – Numerical results for ILS

N	R	J	Obj.	Gap (%)	p ₁	p ₂	p ₃	p ₄	p ₅	p ₆
10	100	2	5.200	0.00	0.626	0.651				
10	100	3	5.142	0.00	0.562	0.652	0.674			
10	100	4	5.140	0.00	0.543	0.560	0.652	0.677		
10	100	5	5.143	0.00	0.543	0.560	0.652	0.677	0.668	
10	100	6	5.155	0.00	0.540	0.576	0.530	0.652	0.656	0.680
10	400	2	5.279	0.00	0.550	0.652				
10	400	3	5.196	0.00	0.550	0.652	0.651			
10	400	4	5.204	0.00	0.549	0.569	0.652	0.657		
10	400	5	5.208	0.00	0.549	0.569	0.652	0.657	0.650	
10	400	6	5.234	0.03	0.564	0.558	0.542	0.650	0.661	0.669