

A Spatial Branch and Bound Algorithm for Continuous Pricing with Advanced Discrete Choice Demand Modeling

Tom Haering Michel Bierlaire

Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
Ecole Polytechnique Fédérale de Lausanne

May 11, 2023



A Spatial Branch and Benders Decomposition Algorithm for Continuous Pricing with Advanced Discrete Choice Demand Modeling

Tom Haering Michel Bierlaire

Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
Ecole Polytechnique Fédérale de Lausanne

May 11, 2023



Outline

1 Introduction

2 Methodology

3 Experimental Results

4 Conclusions



The Continuous Pricing Problem (CPP)

CPP

- Supplier offers S products for sale. Goal: determine optimal **price** for each product to maximize total **profit**.
- Demand for each product is modeled using a **discrete choice model** (DCM).

DCM

- For every **costumer** n and **product** i a stochastic **utility** U_{in} is defined, which depends on **socio-economic** characteristics of the individual and **attributes** of the products (e.g. the price).



The Continuous Pricing Problem (CPP)

Probability

- **Probability** that costumer n chooses alternative i :

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

- **Logit** ($\varepsilon_{in} \sim \text{i.i.d. Gumbel}(0, 1)$):

$$P_n(i) = \frac{e^{V_{in}}}{\sum_{j \in C_n} e^{V_{jn}}}$$

- **Mixed Logit** (Logit + $\beta_k \sim F(\beta_k | \theta)$):

$$P_n(i) = \int \frac{e^{V_{in}(\beta_{kn})}}{\sum_{j \in C_n} e^{V_{jn}(\beta_{kn})}} f(\beta_k | \theta) d\beta_k$$

Monte Carlo Simulation [Panque et al., 2021]

- **Simulate** R scenarios (draws), each with **deterministic** utilities U_{inr} .
- **Choice** variables:

$$\omega_{inr} = \begin{cases} 1 & \text{if } U_{inr} = \max_j U_{jnr} \\ 0 & \text{else} \end{cases}$$

- Probability **estimator**:

$$\hat{P}_n(i) = \frac{1}{R} \sum_r \omega_{inr}$$

MILP formulation [Panque et al., 2021]

$$\max_{p, \omega, U, H} \frac{1}{R} \sum_r \sum_n \sum_{i \in S} p_i \omega_{inr}$$

s.t.

$$\sum_i \omega_{inr} = 1 \quad \forall n, r \quad (\mu_{nr})$$

$$H_{nr} = \sum_i U_{inr} \omega_{inr} \quad \forall n, r \quad (\zeta_{nr})$$

$$H_{nr} \geq U_{inr} \quad \forall i, n, r \quad (\alpha_{inr})$$

$$U_{inr} = \sum_{k \neq p} \beta_k x_{ink} + \beta_p p_i + \varepsilon_{inr} \quad \forall i, n, r \quad (\kappa_{inr})$$

$$\omega \in \{0, 1\}^{JNR}$$

$$p, U, H \in \mathbb{R}^S, \mathbb{R}^{JNR}, \mathbb{R}^{NR}$$



Literature

[Li and Huh, 2011], [Gallego and Wang, 2014], ...

- Extensive research for **Logit** and **Nested Logit** (NL) integration.

[Li et al., 2019], [Marandi and Lurkin, 2020],
[van de Geer and den Boer, 2022], ...

- Tackled **Mixed Logit** (ML) integration in various ways, e.g. **approximations**, assuming consumer **homogeneity**, or considering only **discrete** probability measures.

[Panque et al., 2018]

- Apply a **Lagrangian** decomposition scheme to speed up the solution of the MILP.
- Limited success.

Outline

1 Introduction

2 Methodology

3 Experimental Results

4 Conclusions



QCQP formulation

$$\max_{p, \omega, U, H} \frac{1}{R} \sum_r \sum_n \sum_{i \in S} p_i \omega_{inr}$$

s.t.

$$\sum_i \omega_{inr} = 1 \quad \forall n, r \quad (\mu_{nr})$$

$$H_{nr} = \sum_i U_{inr} \omega_{inr} \quad \forall n, r \quad (\zeta_{nr})$$

$$H_{nr} \geq U_{inr} \quad \forall i, n, r \quad (\alpha_{inr})$$

$$U_{inr} = \sum_{k \neq p} \beta_k x_{ink} + \beta_p p_i + \varepsilon_{inr} \quad \forall i, n, r \quad (\kappa_{inr})$$

$$\omega \in [0, 1]^{JNR}$$

$$p, U, H \in \mathbb{R}^S, \mathbb{R}^{JNR}, \mathbb{R}^{NR}$$

QCLP formulation

$$\max_{p, \omega, \eta, U, H} \frac{1}{R} \sum_r \sum_n \sum_i \eta_{inr}$$

s.t.

$\sum_i \omega_{inr}$	=	1	$\forall n, r$	(μ_{nr})
H_{nr}	=	$\sum_i (\sum_{k \neq p} \beta_k x_{ink} + \varepsilon_{inr}) \omega_{inr} + \beta_p \eta_{inr}$	$\forall n, r$	(ζ_{nr})
H_{nr}	\geq	U_{inr}	$\forall i, n, r$	(α_{inr})
U_{inr}	=	$\sum_{k \neq p} \beta_k x_{ink} + \beta_p p_i + \varepsilon_{inr}$	$\forall i, n, r$	(κ_{inr})
η_{inr}	=	$p_i \omega_{inr}$	$\forall i \in S, n, r$	(λ_{inr})
ω	\in	$[0, 1]^{JNR}$		
p, η, U, H	\in	$\mathbb{R}^S, \mathbb{R}^{SNR}, \mathbb{R}^{JNR}, \mathbb{R}^{NR}$		

Spatial Branch & Bound (B&B) Algorithm

Relaxation

- Assume reasonable **initial bounds** on price, $p_i \in [p_i^L, p_i^U]$
- **McCormick envelope** for $\eta_{inr} = p_i \omega_{inr}$:

$$\begin{aligned}\eta_{inr} &\geq p_i^L \omega_{inr} && \forall i \in S, n, r && (\lambda_{inr}^1) \\ \eta_{inr} &\geq p_i^U \omega_{inr} + p_i - p_i^U && \forall i \in S, n, r && (\lambda_{inr}^2) \\ \eta_{inr} &\leq p_i^L \omega_{inr} + p_i - p_i^L && \forall i \in S, n, r && (\lambda_{inr}^3) \\ \eta_{inr} &\leq p_i^U \omega_{inr} && \forall i \in S, n, r && (\lambda_{inr}^4)\end{aligned}$$

- Integrality is **preserved** for tight enough bounds

Spatial Branch & Bound (B&B) Algorithm

Branching strategy

- We branch on the price p_i where $\eta_{inr} = p_i \omega_{inr}$ has the **strongest maximal violation**.

Convergence

- Every relaxation provides an **upper bound** on the maximal profit.
- Any solution value for the price gives an immediate feasible solution (**lower bound**) due to integrality.

Custom B&B vs. standard B&B

- We only branch on ***S*** continuous variables, instead of branching on ***SNR*** continuous (QCLP) or ***JNR*** binary (MILP) variables.

Benders Decomposition

- The McCormick relaxation at each node is solved by the use of a **Benders decomposition** (cut generation).
- Highly **separable** subproblems: can be solved for each customer and scenario **independently**.
- Can use fully **disaggregated optimality cuts** (one cut per customer and scenario).

Outline

- 1 Introduction
- 2 Methodology
- 3 Experimental Results
- 4 Conclusions



Case Study

Parking space operator [Ibeas et al., 2014]

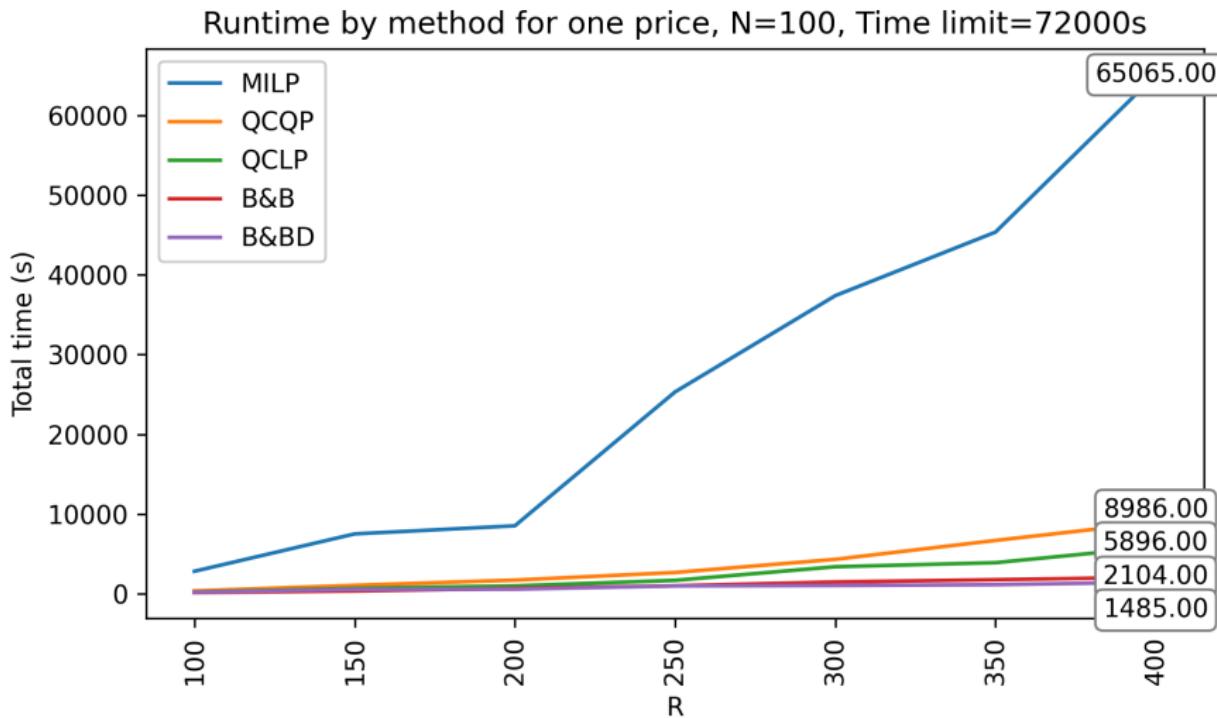
- **Alternatives:** Paid-Street-Parking (PSP), Paid-Underground-Parking (PUP) and Free-Street-Parking (FSP)
- Optimize prices for PSP and PUP, FSP is the **opt-out** alternative
- Choice model is a **Mixed Logit**, $\beta_{\text{fee}}, \beta_{\text{time_parking}} \sim \mathcal{N}(\mu, \sigma)$

Experiments

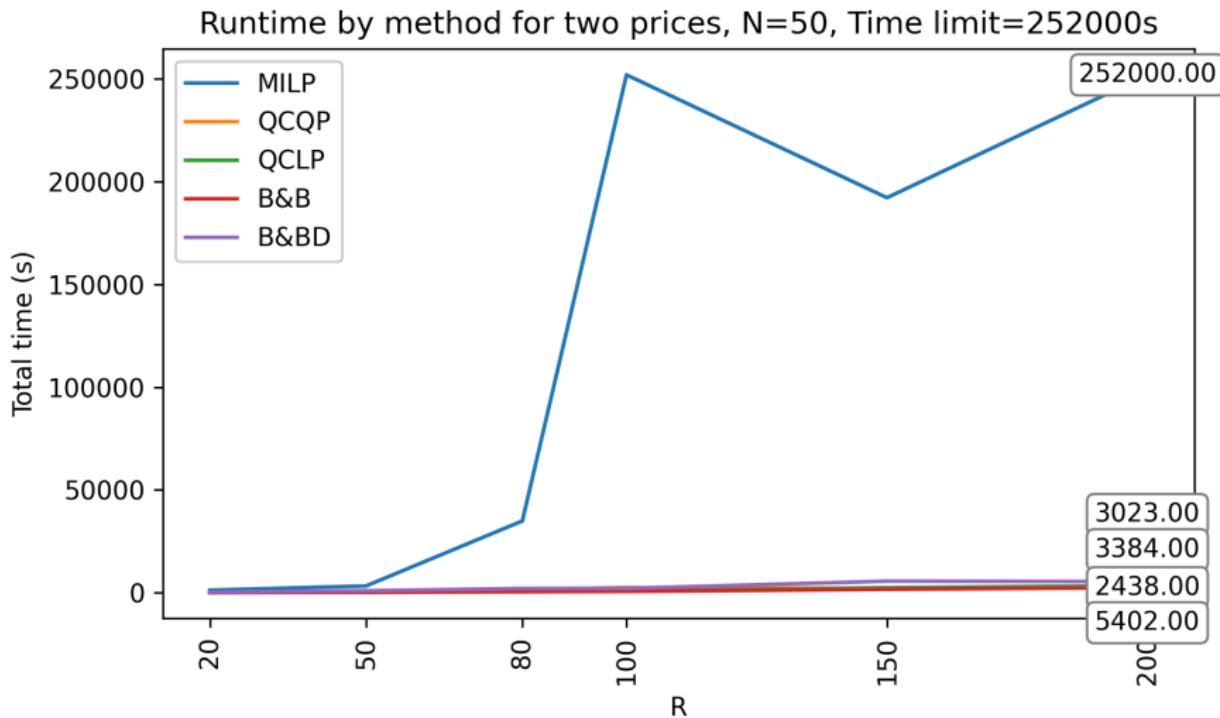
- **One** price vs. **two** prices
- **Small** vs. **large** number of draws



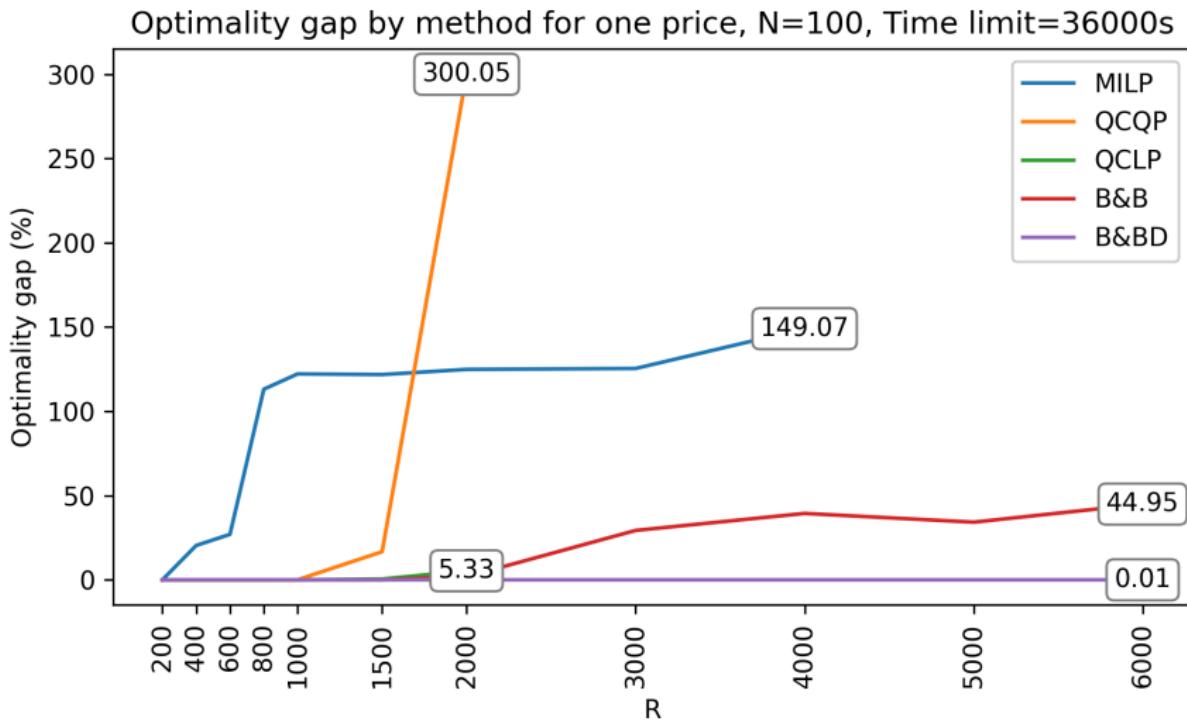
Computational results



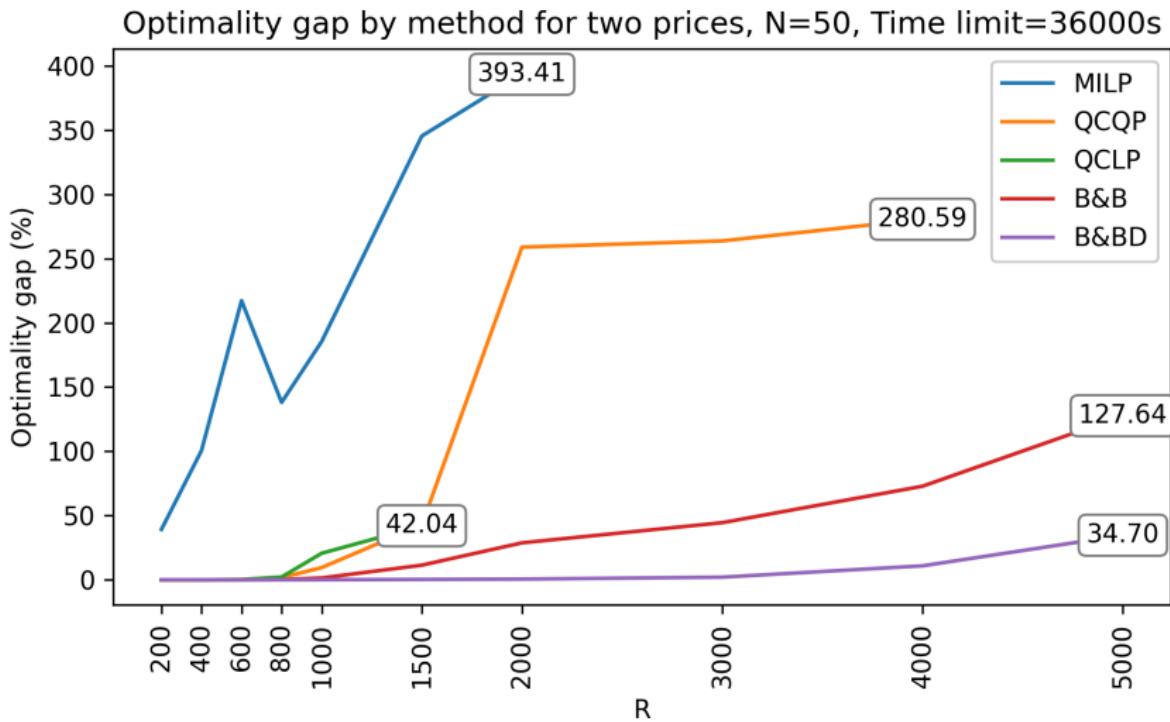
Computational results



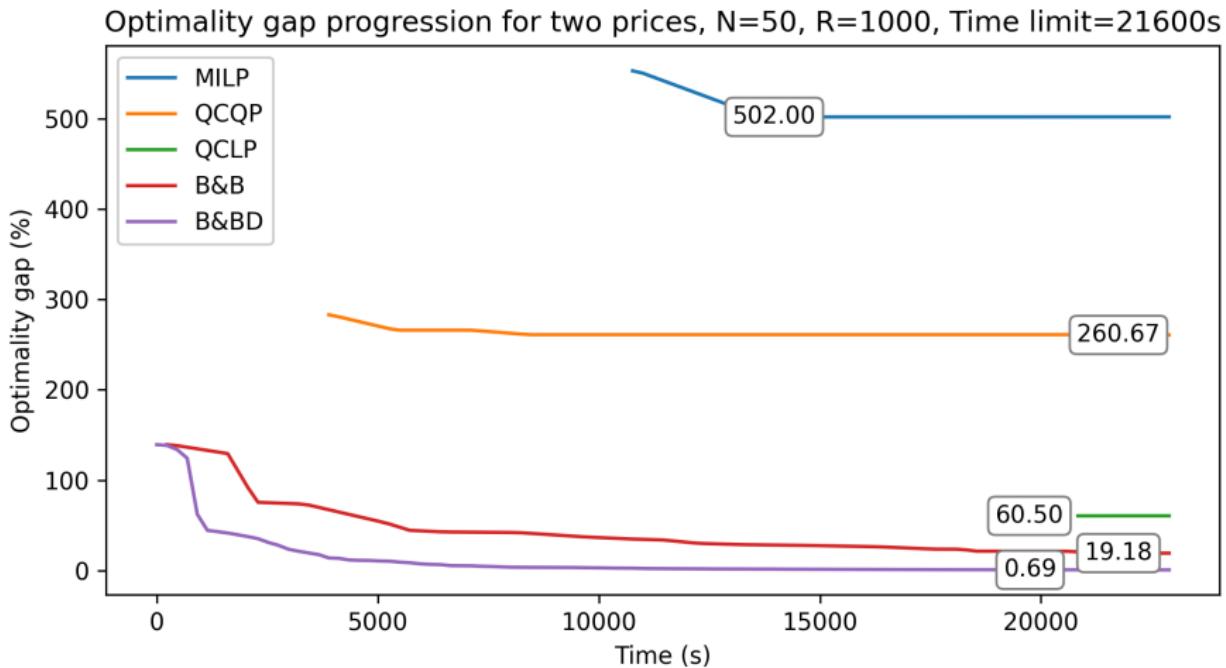
Computational results



Computational results



Computational results



Outline

1 Introduction

2 Methodology

3 Experimental Results

4 Conclusions



Conclusions

- Introduced more efficient **formulation** of the CPP as a QCQP and QCLP.
- Developed methodology that is applicable to **any choice-based optimization problem** integrating **any advanced choice model**.
- Showed that we can solve instances to **optimality** before GUROBI finds a feasible solution.
- **Future research:** complicate the CPP (e.g. capacity constraints), apply to other choice-based optimization problems (e.g. maximum likelihood estimation, revenue management)



Thank you for your attention!



Appendix

Table 1: Utility parameters reported in [Ibeas et al., 2014]

Parameter	Value
ASC_{FSP}	0.0
ASC_{PSP}	32.0
ASC_{PUP}	34.0
Fee (€)	$\sim \mathcal{N}(-32.328, 14.168)$
Fee PSP - low income (€)	-10.995
Fee PUP - low income (€)	-13.729
Fee PSP - resident (€)	-11.440
Fee PUP - resident (€)	-10.668
Access time to parking (min)	$\sim \mathcal{N}(-0.788, 1.06)$
Access time to destination (min)	-0.612
Age of vehicle (1/0)	4.037
Origin (1/0)	-5.762

Appendix

Table 2: Solve time (seconds) for single-price optimization (small-scale)

N	R	MILP	QCQP	QCLP	B&B	B&BD
100	100	2849	392	242	174	216
100	150	7534	1087	708	378	574
100	200	8549	1746	1018	701	603
100	250	25333	2698	1713	1032	1012
100	300	37396	4346	3416	1511	1066
100	350	45362	6715	3927	1795	1169
100	400	65065	8986	5896	2104	1485

Appendix

Table 3: Optimal profit and price for single-price optimization (small-scale)

N	R	MILP		QCQP		QCLP		B&B		B&BD	
		Profit	Price								
100	100	54.134	[0.661]	54.134	[0.661]	54.134	[0.661]	54.133	[0.661]	54.133	[0.661]
100	150	54.233	[0.67]	54.233	[0.67]	54.233	[0.67]	54.233	[0.67]	54.232	[0.67]
100	200	54.599	[0.662]	54.599	[0.662]	54.599	[0.662]	54.598	[0.663]	54.596	[0.662]
100	250	54.622	[0.673]	54.622	[0.673]	54.622	[0.673]	54.619	[0.673]	54.618	[0.673]
100	300	54.48	[0.67]	54.48	[0.67]	54.479	[0.67]	54.479	[0.67]	54.478	[0.67]
100	350	54.449	[0.657]	54.448	[0.657]	54.449	[0.657]	54.448	[0.657]	54.447	[0.657]
100	400	54.389	[0.664]	54.389	[0.664]	54.389	[0.664]	54.389	[0.669]	54.388	[0.664]

Appendix

Table 4: Solve time (seconds) for two-price optimization (small-scale)

N	R	MILP		QCQP		QCLP		B&B	
		Time	Gap (%)	Time	Time	Time	Time	Time	Time
50	20	1238	0.01	60	32	32	32	184	
50	50	3275	0.01	487	199	201	933		
50	80	34907	0.01	1516	564	488	2051		
50	100	251466	0.01	2475	843	614	2099		
50	150	192213	0.01	2105	2404	1651	5614		
50	200	252000	23.92	3023	3384	2438	5402		

Appendix

Table 5: Optimal profit and price for two-price optimization (small-scale)

N	R	MILP		QCQP		QCLP		B&B		B&BD	
		Profit	Price								
50	20	27.417	[0.609, 0.653]	27.417	[0.609, 0.653]	27.417	[0.609, 0.653]	27.416	[0.609, 0.653]	27.414	[0.609, 0.653]
50	50	26.71	[0.556, 0.654]	26.71	[0.556, 0.654]	26.71	[0.556, 0.654]	26.71	[0.556, 0.654]	26.707	[0.556, 0.654]
50	80	27.413	[0.57, 0.648]	27.413	[0.57, 0.648]	27.413	[0.57, 0.648]	27.412	[0.57, 0.648]	27.41	[0.57, 0.648]
50	100	27.546	[0.608, 0.704]	27.546	[0.608, 0.704]	27.546	[0.608, 0.704]	27.544	[0.608, 0.704]	27.544	[0.608, 0.704]
50	150	27.29	[0.562, 0.668]	27.289	[0.562, 0.668]	27.29	[0.562, 0.668]	27.29	[0.562, 0.668]	27.288	[0.562, 0.667]
50	200	26.997	[0.546, 0.679]	26.997	[0.546, 0.679]	26.997	[0.546, 0.679]	26.995	[0.546, 0.679]	26.996	[0.546, 0.679]

Appendix

Table 6: Solve time (seconds) for single-price optimization (large-scale)

N	R	MILP		QCQP		QCLP		B&B		B&BD	
		Time	Gap (%)								
100	200	8348	0.01	1059	0.01	698	0.01	310	0.00	409	0.01
100	400	36000	20.39	5013	0.01	3629	0.01	1255	0.01	1050	0.01
100	600	36000	27.0	14796	0.01	10775	0.01	3110	0.01	1707	0.01
100	800	36000	113.12	21626	0.01	15784	0.01	6206	0.01	2444	0.01
100	1000	36000	122.21	36000	0.04	26727	0.01	10007	0.01	3131	0.01
100	1500	36000	121.82	36000	16.69	36000	0.49	22892	0.01	5093	0.01
100	2000	36000	124.91	36000	300.05	36000	5.33	36000	1.88	7341	0.01
100	3000	36000	125.44	36000	-	36000	-	36000	29.33	12396	0.01
100	4000	36000	149.07	36000	-	36000	-	36000	39.42	20990	0.01
100	5000	36000	-	36000	-	36000	-	36000	34.22	28768	0.01
100	6000	36000	-	36000	-	36000	-	36000	44.95	35917	0.01
100	7000	36000	-	36000	-	36000	-	36000	44.88	36000	0.16

Appendix

Table 7: Optimal profit and price for single-price optimization (large-scale)

N	R	MILP		QCQP		QCLP		B&B		B&BD	
		Profit	Price								
100	200	54.599	[0.662]	54.599	[0.662]	54.599	[0.662]	54.598	[0.663]	54.596	[0.662]
100	400	54.385	[0.664]	54.389	[0.664]	54.389	[0.664]	54.389	[0.669]	54.388	[0.664]
100	600	54.019	[0.625]	54.295	[0.667]	54.295	[0.667]	54.295	[0.667]	54.294	[0.667]
100	800	54.319	[0.662]	54.327	[0.653]	54.326	[0.653]	54.325	[0.653]	54.326	[0.653]
100	1000	54.421	[0.663]	54.429	[0.661]	54.429	[0.661]	54.429	[0.661]	54.429	[0.661]
100	1500	54.488	[0.67]	49.33	[0.971]	54.514	[0.654]	54.53	[0.659]	54.529	[0.659]
100	2000	54.511	[0.656]	22.469	[1.379]	53.966	[0.613]	54.54	[0.667]	54.541	[0.666]
100	3000	54.439	[0.664]	-	-	-	-	52.387	[0.801]	54.448	[0.661]
100	4000	54.422	[0.668]	-	-	-	-	51.175	[0.856]	54.428	[0.669]
100	5000	-	-	-	-	-	-	53.144	[0.764]	54.394	[0.661]
100	6000	-	-	-	-	-	-	49.207	[0.971]	54.399	[0.663]
100	7000	-	-	-	-	-	-	49.229	[0.97]	54.41	[0.669]

Appendix

Table 8: Solve time (seconds) for two-price optimization (large-scale)

N	R	MILP		QCQP		QCLP		B&B		B&BD	
		Time	Gap (%)								
50	200	36000	39.22	3098	0.01	3338	0.01	2426	0.01	5498	0.01
50	400	36000	100.58	17774	0.01	23325	0.01	11746	0.01	21838	0.01
50	600	36000	217.26	36000	0.18	36000	0.26	26662	0.01	35367	0.01
50	800	36000	138.07	36000	1.75	36000	2.21	36000	0.16	35938	0.01
50	1000	36000	185.45	36000	9.52	36000	20.68	36000	1.48	36000	0.07
50	1500	36000	345.36	36000	42.8	36000	42.04	36000	11.41	36000	0.32
50	2000	36000	393.41	36000	258.89	36000	-	36000	28.79	36000	0.58
50	3000	36000	-	36000	263.73	36000	-	36000	44.48	36000	2.08
50	4000	36000	-	36000	280.59	36000	-	36000	72.86	36000	10.90
50	5000	36000	-	36000	-	36000	-	36000	127.64	36000	34.70
50	6000	36000	-	36000	-	36000	-	36000	128.44	36000	41.96
50	7000	36000	-	36000	-	36000	-	36000	138.01	36000	51.96

Appendix

Table 9: Optimal profit and price for two-price optimization (large-scale)

N	R	MILP		QCQP		QCLP		B&B		B&BD	
		Profit	Price								
50	200	26.997	[0.546, 0.679]	26.997	[0.546, 0.679]	26.997	[0.546, 0.679]	26.995	[0.546, 0.679]	26.996	[0.546, 0.679]
50	400	21.689	[0.789, 0.956]	27.174	[0.556, 0.665]	27.174	[0.556, 0.665]	27.172	[0.556, 0.665]	27.172	[0.556, 0.665]
50	600	13.801	[1.087, 1.281]	27.243	[0.561, 0.682]	27.245	[0.563, 0.671]	27.246	[0.562, 0.683]	27.246	[0.562, 0.683]
50	800	16.993	[0.877, 1.203]	27.072	[0.578, 0.668]	27.06	[0.559, 0.659]	27.082	[0.573, 0.667]	27.089	[0.574, 0.667]
50	1000	13.987	[1.2, 1.212]	26.968	[0.59, 0.684]	26.208	[0.584, 0.797]	27.012	[0.571, 0.667]	27.031	[0.573, 0.67]
50	1500	10.144	[1.415, 1.485]	26.319	[0.584, 0.799]	26.322	[0.584, 0.799]	26.982	[0.582, 0.698]	27.052	[0.569, 0.667]
50	2000	9.255	[1.239, 1.866]	11.82	[1.199, 1.395]	-	-	26.718	[0.632, 0.712]	27.094	[0.565, 0.661]
50	3000	-	-	11.849	[1.198, 1.397]	-	-	25.983	[0.5, 0.756]	27.144	[0.571, 0.677]
50	4000	-	-	11.844	[1.199, 1.396]	-	-	24.707	[1.242, 0.766]	27.078	[0.582, 0.699]
50	5000	-	-	-	-	-	-	18.988	[1.0, 1.0]	26.012	[0.5, 0.755]
50	6000	-	-	-	-	-	-	18.915	[1.0, 1.0]	25.973	[0.5, 0.757]
50	7000	-	-	-	-	-	-	18.926	[1.0, 1.0]	24.681	[1.231, 0.766]

Appendix: Callback implementation

```
def mycallback(model, where):
    if where == GRB.Callback.MIPNODE:
        status = model.cbGet(GRB.Callback.MIPNODE_STATUS)
        if status == GRB.OPTIMAL:
            sol = model.cbGetNodeRel(model._vars)
            omega, eta = compute_cpp_from_p_parking(sol[0], sol[1])
            mysol = [sol[0], sol[1]] + list(eta.values()) + list(omega.values())
            model.cbSetSolution(model._vars, mysol)
```

Bibliography I

-  Gallego, G. and Wang, R. (2014).
Multiproduct price optimization and competition under the nested logit model with product-differentiated price sensitivities.
Operations Research, 62(2):450–461.
-  Ibeas, A., Dell'Olio, L., Bordagaray, M., and Ortúzar, J. d. D. (2014).
Modelling parking choices considering user heterogeneity.
Transportation Research Part A: Policy and Practice, 70:41–49.
-  Li, H. and Huh, W. T. (2011).
Pricing multiple products with the multinomial logit and nested logit models: Concavity and implications.
Manufacturing & Service Operations Management, 13(4):549–563.

Bibliography II

 Li, H., Webster, S., Mason, N., and Kempf, K. (2019).

Product-line pricing under discrete mixed multinomial logit demand:
winner—2017 m&som practice-based research competition.

Manufacturing & Service Operations Management, 21(1):14–28.

 Marandi, A. and Lurkin, V. (2020).

An exact algorithm for the static pricing problem under discrete mixed
logit demand.

arXiv preprint arXiv:2005.07482.

 Paneque, M. P., Bierlaire, M., Gendron, B., and Azadeh, S. S. (2021).
Integrating advanced discrete choice models in mixed integer linear
optimization.



Transportation Research Part B: Methodological, 146:26–49.



Bibliography III

-  Paneque, M. P., Gendron, B., Lurkin, V., Azadeh, S. S., and Bierlaire, M. (2018).
A lagrangian relaxation technique for the demand-based benefit maximization problem.
In *Proceedings of the 18th Swiss Transport Research Conference (Ascona, Switzerland)*.
-  van de Geer, R. and den Boer, A. V. (2022).
Price optimization under the finite-mixture logit model.
Management Science, 68(10):7480–7496.