Partially-Ranked Choice Models for Data-Driven Assortment Optimization

Sanjay Dominik Jena
Andrea Lodi
Hugo Palmer

Canada Excellence Research Chair, andrea.lodi@polymtl.ca

COW 2018
January 12, 2018, Aussois
Assortment planning: Context

- Process of identifying the set of products that should be offered to the customer
- Direct impact on profit
  - online ads: number of clicks on ads; sales by visiting links, etc.
  - retail: conversion rate of a product, i.e., frequency of sales

Examples:

- Online advertising
- Brick-and-mortar retail
Assortment planning: Objectives

- Find assortment that maximizes revenue
- Encourage the user to select the product(s) that has/have highest utility (e.g. profit)
- In retail: assortment changes can be quite costly

Examples:

- Online advertising
- Brick-and-mortar retail
Assortment Planning: Challenges

- Small assortments $\implies$ less choice $\implies$ less sales!
- More products $\implies$ more choice $\implies$ more sales?
  - offering all products is known to be non-optimal
- Substitution effect
  - the presence of a product may jeopardize the sales of another
  - e.g. the Apple iPad reduced the sales of the Apple Powerbook
  - the absence of a preferred product may encourage the customer to “substitute” to a (more profitable) alternative
- Complexity of assortment constraints:
  - capacity: limited shelf size or space on website
  - product dependencies: subset constraints, balance between product categories (e.g. male and female shoe models) etc.
Assortment Planning: Challenges

Given historical data on assortments and transactions:

How to learn from historical transaction data to predict the performance of a future assortment?

→ customer choice models
Parametric Choice Models

Multinomial Logit (MNL) models
- Attributes an utility to each product
- The probability that a customer selects product $i$ from assortment $S$ is: $P(i|S) = (e^{u_i})/(e^{u_0} + \sum_{j \in S} e^{u_j})$
- Independence of Irrelevant Alternatives (IIA) property
  - Cannot capture substitution effect

Nested Logit (NL) models
- capture certain substitution among categories, but each nest is subject to the IIA property

Mixed Multinomial Logit (MMNL) models
- Overcomes shortfalls of MNL and NL models
- Computationally expensive; overfitting issues
Rank-based choice models

Customer behavior $\sigma_k$: list of products ranked according to preferences of customer $k$, e.g. (2, 4, 0, 1, 3, 5, 6):

Customer selects highest ranked product in the assortment.

Choice model: composed of behaviors $\sigma$ and corresponding probabilities $\lambda_k$ that a random customer follows behavior $\sigma_k$. 
Recent approaches using rank-based choice models

Challenge: an N-factorial large search space of customer behaviors

- Honhon et al. (2012), Vulcano and Van Ryzin (2017), etc.
  - require market knowledge, e.g. customer behaviors
- Jagabathula (2011) and Farias et al. (2013)
  - find the worst-case choice model for a given assortment
  - tractable approach to estimate probabilities for all behaviors
  - find the sparsest model
- Bertsimas and Misic (2016)
  - master problem minimizes estimation error for given behaviors
  - column generation to find new customer behaviors
  - pricing problem solved heuristically, since exact MIP intractable
  - limited to small number of products
Scope and Objectives of this work

Objectives:

- Develop an (efficient) data-driven approach to design optimized assortments
- Consider substitution effect (cannibalization)
- Integrate complex side constraints on the assortment (size, precedence, etc.)
- Be easy to interpret and provide market insights to management: \textit{sparse} and \textit{concise} models
Scope and Objectives of this work

Objectives:

▶ Develop an (efficient) data-driven approach to design optimized assortments
▶ Consider substitution effect (cannibalization)
▶ Integrate complex side constraints on the assortment (size, precedence, etc.)
▶ Be easy to interpret and provide market insights to management: sparse and concise models

Industrial collaboration:

▶ JDA Labs (research lab of JDA Software)
▶ Data from a large North-American retail chain
  ▶ clothes (shoes and shirts)
  ▶ seasonal choice of products
Partially-Ranked Choice Models with Indifference Sets

A new choice model:

- The customer has a strict preference on certain products.
- If unavailable, the customer may buy any *similar* product, which is available, without preference.
Partially-Ranked Choice Models with Indifference Sets

A new choice model:

- The customer has a strict preference on certain products.
- If unavailable, the customer may buy any similar product, which is available, without preference.

Consider a customer behavior \((P(\sigma), I(\sigma), 0)\), e.g. \((3, 4, 1, \{2, 5, 6\}, 0)\)

- \(P(\sigma) = (3, 4, 1) \subseteq \mathcal{N}\) is a strictly ranked list of preferred products
- \(I(\sigma) = \{2, 5, 6\} \subseteq \mathcal{N} \setminus P(\sigma)\) is the subset of indifferent products which will be chosen with uniform probability
Partially-Ranked Choice Models: Properties

(I) Equivalence of choice models:

- Transformation from fully-ranked \((\sigma_C, \lambda_C)\) to partially-ranked choice model \((\sigma_P, \lambda_P)\), and vice versa.

- Partially-ranked behaviors more compact: factorially large number of fully-ranked behaviors required to represent the same buying behavior.
Partially-Ranked Choice Models: Properties

(I) Equivalence of choice models:
- Transformation from fully-ranked \((\sigma_C, \lambda_C)\) to partially-ranked choice model \((\sigma_P, \lambda_P)\), and vice versa.
- Partially-ranked behaviors more compact: factorially large number of fully-ranked behaviors required to represent the same buying behavior.

(II) (Ir)relevance of low ranked products:
- Low ranked products → less important & explain less sales
  - e.g. in assortment density 0.5, the probability that product at rank 10 is selected from an “average” assortment is 0.05%.
- Explanatory power of indifference sets in “average” assortment is similarly low.
- → concise list of strictly ranked products → insights for managers.
Simplified Partially-Ranked Choice Model

Given:

- equal transformation: partial to completely ranked behaviors
- irrelevance of low ranked products and the likely small impact of indifference set on explaining the sales

we consider a **simplified** variant:

\[(P(\sigma), I(\sigma), 0)\]

where:

- \(P(\sigma) = (3, 4, 1) \subseteq \mathcal{N}\) is a strictly ranked list of preferred products
- \(I(\sigma) = \mathcal{N} \setminus P(\sigma) = \{0, 2, 5, 6\}\) is the indifference set.

\(\implies\) several computational advantages without compromising theoretical coherence
Training and Testing the Choice Models

Training set

- Set of $M$ assortments: $\{S_m\}, m = 1, \ldots, M$
- Probabilities of selling product $i$ in assortment $S_m$ to a random customer: $(v_{i,m})$
Training and Testing the Choice Models

Training set
- Set of $M$ assortments: $\{S_m\}, m = 1, \ldots, M$
- Probabilities of selling product $i$ in assortment $S_m$ to a random customer: $(v_{i,m})$

Test set
- Sales for each product $i$ in each of the $M$ other assortments
Training the choice model: completely ranked behaviors

- Given: a subset of customer behaviors and historical sales \( \mathbf{v} \)
- Find: probability distribution \( (\lambda) \) that best explains the sales
- Define a choice matrix \( \mathbf{A} \), for each behavior \( k \) and product/assortment tuple \((i, m)\) (BM, 2016):

\[
A_{i,m}^k = \begin{cases} 
1 & \text{if } i \text{ is chosen by customer } k \text{ in assortment } S_m \\
0 & \text{if } i \text{ is not chosen by customer } k \text{ in assortment } S_m
\end{cases}
\]
Training the choice model: completely ranked behaviors

- Given: a subset of customer behaviors and historical sales \( \mathbf{v} \)
- Find: probability distribution \((\mathbf{\lambda})\) that best explains the sales
- Define a choice matrix \( \mathbf{A} \), for each behavior \( k \) and product/assortment tuple \((i, m)\) (BM, 2016):

\[
A^k_{i,m} = \begin{cases} 
1 & \text{if } i \text{ is chosen by customer } k \text{ in assortment } S_m \\
0 & \text{if } i \text{ is not chosen by customer } k \text{ in assortment } S_m 
\end{cases}
\]

Linear program to find \( \mathbf{\lambda} \) that minimizes estimation error

\[
\begin{align*}
\min_{\mathbf{\lambda}, \mathbf{\epsilon}^+, \mathbf{\epsilon}^-} & \quad 1^T \mathbf{\epsilon}^+ + 1^T \mathbf{\epsilon}^- \\
\text{s.t.} & \quad \mathbf{A} \mathbf{\lambda} + \mathbf{\epsilon}^+ - \mathbf{\epsilon}^- = \mathbf{v} \quad (\alpha) \\
& \quad 1^T \mathbf{\lambda} = 1 \quad (\nu) \\
& \quad \mathbf{\lambda}, \mathbf{\epsilon}^+, \mathbf{\epsilon}^- \geq 0
\end{align*}
\]
Training the choice model: partially-ranked behaviors

Required: the total probability of buying a product sums to 1:

\[ \forall (k, m), \sum_i A^k_{i,m} = 1 \]
Training the choice model: partially-ranked behaviors

Required: the total probability of buying a product sums to 1:

$$\forall (k, m), \sum_{i} A^k_{i, m} = 1$$

For partially-ranked behaviors, a term $$\frac{1}{|S_m|}$$ is distributed on the products in the indifference set:

$$A^k_{i, m} = \begin{cases} 
1, & \text{if } i \text{ is chosen by customer } k \text{ among assortment } S_m \\
0, & \text{if } i \text{ is not chosen by customer } k \text{ among assortment } S_m \\
\frac{1}{|S_m|}, & \text{if } i \in S_m, i \in I(\sigma_k) \text{ and } P(\sigma_k) \cap S_m = \emptyset 
\end{cases}$$
How to efficiently find important behaviors?

- BM (2016) use column generation to find columns $k$ for the LP above
  - MIP pricing problem is intractable for large instances; local search converges slowly
How to efficiently find important behaviors?

- BM (2016) use column generation to find columns $k$ for the LP above
  - MIP pricing problem is intractable for large instances; local search converges slowly

Questions:
- How to explore special structure of indifference sets?
- How to explore the fact that high ranked products have much more impact?
  - low-ranked products may eventually not be considered
How to efficiently find important behaviors?

- BM (2016) use column generation to find columns $k$ for the LP above
  - MIP pricing problem is intractable for large instances; local search converges slowly

Questions:

- How to explore special structure of indifference sets?
- How to explore the fact that high ranked products have much more impact?
  - low-ranked products may eventually not be considered

$\Rightarrow$ expansion of a tree: each node represents a behavior.
$\Rightarrow$ Growing Decision Tree (GDT)
Expansion of the Growing Decision Tree (GDT)

Iteration 1

Tree is initialized with $N$ behaviors: one for each product:

- $P(\sigma_1) = (0)$, $I(\sigma_1) = \emptyset$
- $P(\sigma_2) = (1)$, $I(\sigma_2) = N \setminus 1 = \{0, 2, 3\}$
- $P(\sigma_3) = (2)$, $I(\sigma_3) = N \setminus 2 = \{0, 1, 3\}$
- $P(\sigma_4) = (3)$, $I(\sigma_4) = N \setminus 3 = \{0, 1, 2\}$

Master problem is solved.
Expansion of the Growing Decision Tree (GDT)

Iteration 1

For each of the relevant customer behaviors $\sigma_k$:

- compute reduced costs (using dual values from Master problem)
Expansion of the Growing Decision Tree (GDT)

Iteration 2

- add customer behaviors with lowest (negative) reduced costs to the Master problem
- resolve Master problem
Expansion of the Growing Decision Tree (GDT)

Iteration 2

```
1) λ1
   2) λ5
      1) λ2
         2) rc1
         3) rc2
       3) λ6
          1) λ7
             2) rc6
             3) rc7
       2) λ3
          3) λ4
             1) rc8
             3) rc9
```
Expansion of the Growing Decision Tree (GDT)

Iteration 3
Problem Instances

Randomly generated instances:

- Via Mixed Multinomial Logit model: $K$ classes (one for each customer type)
- Uniformly $[0, 1]$ chosen utilities for all products
- Random selection of 4 products: 100 times higher utilities
- $M$ (typically $= 40$) assortments (20 to train, 20 to test)
Problem Instances

Randomly generated instances:
- Via Mixed Multinomial Logit model: $K$ classes (one for each customer type)
- Uniformly $[0, 1]$ chosen utilities for all products
- Random selection of 4 products: 100 times higher utilities
- $M$ (typically $= 40$) assortments (20 to train, 20 to test)

Industrial data:
- From JDA Labs: Northamerican retail chain (shoes)
- 10 stores during 10 consecutive weeks $\implies$ 100 assortments
- 192 products
Generated data - Learning curves

\[ n = 100 \text{ products} \]

Learning curves: training and test error CG-GDT and CG-LS (BM)
Real industrial data - Computational results

\( n = 192 \) products

Learning curves: training and test error CG-GDT and CG-LS (BM)
Training phase: Scalability & Sparsity

Learning performance and generated choice models sizes $K$ for:
- CG-GDT
- CG-LS

Averaged over 10 random instances
$M = 20$, $\epsilon_0 = 0.01$
Assortment density = 0.3 (assortment size equals $0.3 \times N$)
24 hours time limit, 48 Gbyte memory limit

<table>
<thead>
<tr>
<th>$N$</th>
<th>Train. error</th>
<th>CG-GDT time (sec)</th>
<th># iter</th>
<th># inst. oom</th>
<th>Train. error</th>
<th>CG-LS time (sec)</th>
<th># iter</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.37</td>
<td>2.3</td>
<td>9.2</td>
<td>0</td>
<td>0.39</td>
<td>22.5</td>
<td>392.0</td>
<td>223.8</td>
</tr>
<tr>
<td>50</td>
<td>0.38</td>
<td>6.0</td>
<td>10.3</td>
<td>0</td>
<td>0.40</td>
<td>57.3</td>
<td>603.2</td>
<td>370.1</td>
</tr>
<tr>
<td>100</td>
<td>0.39</td>
<td>29.7</td>
<td>15.4</td>
<td>0</td>
<td>0.40</td>
<td>269.8</td>
<td>1,070.7</td>
<td>721.3</td>
</tr>
<tr>
<td>250</td>
<td>0.39</td>
<td>321.8</td>
<td>21.0</td>
<td>0</td>
<td>0.40</td>
<td>5,204.8</td>
<td>2,492.9</td>
<td>1,788.7</td>
</tr>
<tr>
<td>500</td>
<td>0.38</td>
<td>2,341.5</td>
<td>19.4</td>
<td>1</td>
<td>0.40</td>
<td>49,615.3</td>
<td>4,555.0</td>
<td>3,484.2</td>
</tr>
<tr>
<td>1000</td>
<td>0.33</td>
<td>5,511.2</td>
<td>7.0</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Results for assortment densities 0.1 and 0.5 show the same tendencies.
Choice model: concision

Characteristics of the generated choice model
Averaged over 10 random instances
$M = 20$, $\epsilon_0 = 0.01$
Assortment density = 0.3 (assortment size equals $0.3 \times N$)
720 minutes time limit, 48 Gbyte memory limit

<table>
<thead>
<tr>
<th>$\epsilon_0$</th>
<th>$N$</th>
<th># iter</th>
<th>$K$</th>
<th># strictly ranked products</th>
<th>% explained by indifference sets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>avg</td>
<td>max</td>
</tr>
<tr>
<td>0.01</td>
<td>30</td>
<td>10.2</td>
<td>105.6</td>
<td>2.24</td>
<td>4</td>
</tr>
<tr>
<td>0.01</td>
<td>50</td>
<td>11.3</td>
<td>104.7</td>
<td>1.84</td>
<td>4</td>
</tr>
<tr>
<td>0.01</td>
<td>100</td>
<td>16.4</td>
<td>127.3</td>
<td>1.55</td>
<td>3</td>
</tr>
<tr>
<td>0.01</td>
<td>250</td>
<td>22.0</td>
<td>213.3</td>
<td>1.22</td>
<td>3</td>
</tr>
<tr>
<td>0.01</td>
<td>500</td>
<td>20.4</td>
<td>416.6</td>
<td>1.07</td>
<td>3</td>
</tr>
<tr>
<td>0.01</td>
<td>1000</td>
<td>8.8</td>
<td>836.2</td>
<td>1.03</td>
<td>2</td>
</tr>
<tr>
<td>0.01</td>
<td>all</td>
<td>14.9</td>
<td>300.6</td>
<td>1.49</td>
<td>4</td>
</tr>
</tbody>
</table>
Assortment optimization

Given a choice model, which subset of the products is likely to maximize the revenue?

Literature

- Problem **NP-hard** ($2^n$ revenues to compute by explicit enumeration).
- *If all prices are equal*: Mahajan & van Ryzin (1999) have proposed a linear-complexity algorithm.
- General case: only heuristics (see for example *ADXOpt* by Jagabathula (2011).
- Parametric choice models generally lead to difficult formulations for assortment optimization.
Assortment Optimization: Mixed Integer Programming

Completely ranked preference lists:

- Efficient MIP to find optimal assortment (BM, 2016)
- MIP requires completely ranked customer behaviors
**Assortment Optimization: Mixed Integer Programming**

Completely ranked preference lists:
- Efficient MIP to find optimal assortment (BM, 2016)
- MIP requires completely ranked customer behaviors

Partially-ranked lists from GDT:
- (a) boosting: remaining ranks can be completed at random
- (b) add "indifference constraints":
  - If strictly ranked products are not in the assortment: distribute sales flow (1 unit) uniformly on all products in the indifference set that are part of the assortment
  - forces all products with equal rank to take same values
  - $K \times N^2$ constraints $\implies$ branch-and-cut
Assortment optimization: Scalability

Scalability of assortment optimization for:

- CG-GDT with AO B&C
- CG-GDT with AO-Boosting
- CG-LS with classical AO-MIP

Averaged over 10 random instances
720 minutes time limit, 48 Gbyte memory limit

<table>
<thead>
<tr>
<th>N</th>
<th>CG-GDT with AO B&amp;C</th>
<th>CG-GDT with AO-Boosting</th>
<th>CG-LS with classical AO-MIP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>time (min)</td>
<td>GT revenue</td>
</tr>
<tr>
<td>30</td>
<td>109.9</td>
<td>0.1</td>
<td>74.5</td>
</tr>
<tr>
<td>50</td>
<td>113.5</td>
<td>0.1</td>
<td>82.5</td>
</tr>
<tr>
<td>100</td>
<td>117.8</td>
<td>0.8</td>
<td>88.8</td>
</tr>
<tr>
<td>250</td>
<td>211.0</td>
<td>7.3</td>
<td>90.4</td>
</tr>
<tr>
<td>500</td>
<td>438.1</td>
<td>113.1</td>
<td>94.5</td>
</tr>
<tr>
<td>1000</td>
<td>897.4</td>
<td>669.9</td>
<td>95.0</td>
</tr>
<tr>
<td>all</td>
<td>314.6</td>
<td>131.9</td>
<td>87.6</td>
</tr>
</tbody>
</table>

Revenue: value based on ground-truth MMNL model
Boosting: at least 3 randomly completed lists for each k
Summary

- New representation for rank-based choice models
  - Indifference sets
  - Implicitly equivalent to choice models with completely ranked behaviors
Summary

- New representation for rank-based choice models
  - Indifference sets
  - Implicitly equivalent to choice models with completely ranked behaviors
- Computational advantages
  - Fast training of choice model; good convergence after few iterations
  - Fast generation of new customer behaviors (products with high ranks have more impact)
Summary

- New representation for rank-based choice models
  - Indifference sets
  - Implicitly equivalent to choice models with completely ranked behaviors
- Computational advantages
  - Fast training of choice model; good convergence after few iterations
  - Fast generation of new customer behaviors (products with high ranks have more impact)
- Advantages from the managerial perspective
  - Model is sparse: less customer behaviors
  - Model is concise: low number strictly ranked products
Open research directions

Extensions:

- Learn the choice model by “classical” ML algorithms
- Generalization to new products: how can we learn the importance of products that have never been part of past assortments?

Q (?) & A (!)
References

Noisy Data: Example

Assortment 1:
- contains products \{1, 2, 3, 4\}
- products 1 and 2 have been sold

Assortment 2:
- contains products \{1, 2, 3, 4, 5\}
- products 3 and 4 have been sold

⇒ When training on assortment 1 and testing on assortment 2, sales **cannot** be captured by a rank-based customer behavior.
Column generation: identifying relevant columns

**Reduced cost**

- **Reduced cost** of a new column $a$ of $A$: $rc(a) = -\alpha a - \nu$
- $rc(a) < 0$ means that adding $a$ to the matrix $A$ is likely to improve the objective value (the smaller, the better!)

---

**General algorithm**

1. Choose randomly a customer’s behavior
2. Solve the master problem and get the dual variables $\alpha$ and $\nu$
3. Find a new column achieving a negative reduced cost
4. Add this new column to $A$ and go to 2.
Partially-Ranked Choice Models for Data-Driven Assortment Optimization

---

**Conclusion**

GDT Algorithm 1/2

**Algorithm 1:** GDT-based column generation algorithm

**Input:**
- Sales probability vector \( v \), training set of assortments \( S_1, \ldots, S_M \).
- Maximum number of column generation iterations \( \text{iter}^{CG}_{MAX} \).
- Optimality training criteria \( \epsilon_{MIN}^{Tr} \).
- Maximum number of sub-behaviors \( \delta \) that are added at each iteration.
- Maximum depth \( \tilde{d} \) to find sub-behaviors with negative reduced costs.

**Output:**
- A set \( K = \{\sigma_0, \ldots, \sigma_{K-1}\} \) of \( K \) customer behaviors, where \( \sigma_k = (P(\sigma_k), I(\sigma_k), 0) \).

1. begin
2. Initialize set \( K \) with \( N + 1 \) sub-behaviors \( \sigma_k \), \( k = 0, \ldots, N \), defined with \( P(\sigma_k) = (k) \) and \( I(\sigma_k) = N \setminus \{k\} \).
3. Set \( \text{iter} \) to 0.
4. Solve restricted Master Problem (1) to obtain \( \lambda, \epsilon^+, \epsilon^- \) and dual values \( \alpha \) and \( \nu \).
5. while \((\text{iter} \geq \text{iter}^{CG}_{MAX}) \) and \((1^T\epsilon^+ + 1^T\epsilon^- > \epsilon_{MIN}^{Tr})\) do
6.  Set \( \text{iter} \leftarrow \text{iter} + 1 \)
7.  Set \( \text{depth} \leftarrow 0 \)
8.  Set \( C^P \leftarrow K \)
9.  Set \( C^N \leftarrow \emptyset \)
10. Set \( \mathcal{D} \) to \( \emptyset \).
11. while \((\mathcal{D} = \emptyset) \) and \((\text{depth} \leq \tilde{d})\) do
12.    Set \( \text{depth} \leftarrow \text{depth} + 1 \)
GDT Algorithm 2/2

13  for \( \forall \sigma_k \in C^P \) do
14      if \( \mathcal{P}(\sigma_k)|_{\mathcal{P}(\sigma_k)} \neq 0 \) (i.e., if last element in \( \mathcal{P}(\sigma_k) \) is not 0) then
15          compute the reduced costs for all new sub-behaviors of \( \sigma_k \).
16          Add each new sub-behavior with negative reduced costs to \( C^N \).
17  if \( (C^N = \emptyset) \) then
18      Set \( C' \leftarrow \emptyset \)
19      for \( \forall \sigma_k \in C^P \) do
20          Add all sub-behaviors of \( \sigma_k \) to \( C' \)
21          Set \( C^P \leftarrow C' \)
22  else
23      Add to \( D \) up to \( \delta \) sub-behaviors \( \sigma_k \in C^N \) which have the lowest reduced costs.
24  if \( (D = \emptyset) \) then
25      Run MIP to find \( \sigma_k \) with most negative reduced cost
26      if (\( \sigma_k \)'s reduced costs are negative) then
27          Add \( \sigma_k \) to \( D \)
28      else
29          Return \( K \)
30  Set \( K = K \cup D \) and all sub-behaviors \( \in D \) as new columns to matrix \( A \).
31  Solve restricted Master Problem (1) to obtain \( \lambda, \epsilon^+, \epsilon^- \) and dual values \( \alpha \) and \( \nu \).
32  return \( K \);
## AO Mixed Integer Program: Completely ranked behaviors

### Assortment Optimization

Efficiently finds optimal assortment (BM, 2016)

- $x_i = 1$: $i$ in optimal assortment
- $y_i^k = 1$: $i$ chosen by customer's behavior $k$ at optimality

### MIP

\[
\begin{align*}
\max_{x, y} & \quad \sum_{k=1}^{K} \sum_{i=1}^{n} r_i \lambda^k y_i^k \\
\text{s.t.} & \quad \sum_{i=0}^{n} y_i^k = 1 \quad \forall k \\
& \quad y_i^k \leq x_i \quad \forall (i, k) \\
& \quad \sum_{j: \sigma^k(j) > \sigma^k(i)} y_j^k \leq 1 - x_i \quad \forall (i, k) \\
& \quad \sum_{j: \sigma^k(j) > \sigma^k(0)} y_j^k = 0 \quad \forall k \\
& \quad x_i \in \{0, 1\}, y_i^k \geq 0
\end{align*}
\]

- MIP requires completely ranked customer behaviors
- Behaviors generated by the Growing Decision Tree are partially-ranked
- ⇒ remaining ranks can be completed at random (boosting)
AO Mixed Integer Program: Partially-ranked behaviors

Extend the previous MIP to directly work on choice model with partially-ranked customer behaviors

- If strictly ranked products are not in the assortment:
  - Distribute sales flow (1 unit) uniformly on all products in the indifference set that are part of the assortment
  - $\implies$ force all products with equal rank to take same values

- Add following $(K \times N^2)$ constraints to previous MIP:

  $y_i^k - y_j^k \leq 2 - x_i - x_j \quad \forall k \in \{1, \ldots, K\}; \quad \forall i, j \in \{1, \ldots, N\} : i > j$ and $\sigma^k(i) = \sigma^k(j)$

  $- y_i^k + y_j^k \leq 2 - x_i - x_j \quad \forall k \in \{1, \ldots, K\}; \quad \forall i, j \in \{1, \ldots, N\} : i > j$ and $\sigma^k(i) = \sigma^k(j)$

$\implies$ add via branch-and-cut
Choice model: concision

Example probl.: 30 products, 20 assortments, assortment size: 15

Evaluation of model sparsity:

<table>
<thead>
<tr>
<th>minimum ( \lambda ) value</th>
<th># behaviors CG-LS</th>
<th># behaviors CG-GDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5%</td>
<td>63</td>
<td>52</td>
</tr>
<tr>
<td>0.1%</td>
<td>182</td>
<td>114</td>
</tr>
<tr>
<td>0.0%</td>
<td>225</td>
<td>123</td>
</tr>
</tbody>
</table>

Evaluation of concision (number of strictly ranked products in behavior):

<table>
<thead>
<tr>
<th># of products in preference list</th>
<th># lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>72</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>