ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

Master Thesis in Econometrics & Management Science

Business Analytics & Quantitative Marketing

Conjoint Design Generation On-The-Fly

Supervisor
Prof. dr.
Dennis FOK

Co-Supervisor
Prof. dr.
Patrick J.F. GROENEN

Candidate
Adrian MARTINEZ DE LA TORRE
ID: 451028

SEPTEMBER 2017

Abstract

Many companies and marketers need to decide which features and characteristics to include in a product. This research develops a novel algorithm for Choice Based Conjoint Analysis On-The-Fly (CBC-OTF). The goal of CBC-OTF is to find a global optimal alternative among thousands or even millions of possible products with respect to an individual and population level criteria. We iteratively drop uninformative levels, features, from the design in order to quickly converge to the best configuration. A novel approach using Fedorov algorithm for faster and more efficient designs is used. By using an aggregated logit model we manage to get utility part-worths for the non-removed levels with very low computational costs and high precision. Repeated simulations show that the candidate global optimum reached deviates on average less than 15% from the individual optimal product. If a standard approach is used i.e. CBC with no level removal, the deviation increases to 35%. If optimality at population level is considered, precision is not very high, 35% of predictions are correct.

Keywords: Conjoint Analysis, Design generation, CBC, D-Efficiency

JEL: C15, C39, C9, M3

Acknowledgements

I would like to thank the opportunity given by SKIM to write this Thesis and be part of their team during the last six months. I am very grateful to Remco Don, Senior Research Analyst at SKIM, who helped me throughout the research and writing with his interesting and savvy insights. Special mention to Prof. Dr. Dennis Fok who gave me very useful guidelines to successfully complete this Thesis. I also want to thank Prof. Dr. Patrick J.F. Groenen for his contribution and corrections as Co-Supervisor. Finally, important people to mention who contributed to some extent in the elaboration of this Thesis are Kees van der Wagt, Jeroen Hardon and Marco Hoogerbrugge.

Contents

1 Introduction

Many companies usually face the problem of deciding which product they should produce and commercialize. This is not a trivial question, the amount of sales will depend on how the product is positioned in the market and how well aligned it is with consumer preferences. The selection of the adequate product, in terms of features and characteristics, becomes paramount.

Any product in the marketplace is conformed by a different set of features which can take many different forms and shapes; for instance, the color of a soda can, the level of sugar, the font of the label etcetera. But which is the right combination of features of a product that make it the most attractive to the great majority of consumers? This research presents a new methodology to give an answer to this frequent issue managers face.

The usual approach among marketers and market research agencies is based on Conjoint Analysis. This is a statistical technique that aims to retrieve valuation of the different characteristics of a product by asking people to rate them. Recent developments in the field have gone in the line of using Choice-Based Conjoint Analysis (CBC) as main method to retrieve information on the valuation of features. A main goal is to retrieve the exact utility value of all attributes. This technique consists of showing respondents product profiles, that is, feasible combinations of attributes that shape a product, and asking them to select their preferred one among several different profiles. This approach has some advantages: it is integrated in the most important statistical softwares like SSI and SAS, it is a straightforward procedure from a methodological standpoint and it allows one to estimate interaction between attributes. However, it also suffers from some important shortcomings: There is no actual learning during the data collection process. This in turn leads to less insightful utility estimates because the tasks presented are not improved during the survey as we do not use the information that respondents provide. Furthermore, traditional CBC requires large amounts of respondents to get precise estimates given that there are many trade-offs to evaluate. This means increased economic costs to undertake a traditional CBC study. In addition, the fact that it is part of many important statistical software packages can become a problem in practice: there is little room for customization, meaning that very little improvement can be done in the existing code.

This research proposes a novel approach to conduct CBC studies by using past respondents information. The objective is to reduce the space of alternatives iteratively up to the point that the most valued alternative is left. Traditional CBC studies generate a design and ask to a large fixed amount of respondents so as to obtain utility part-worths per level. Our proposal consists of conducting several CBC studies each with few respondents, and on each iteration removing the least informative levels from the design. This way we can maximize the information retrieved from respondents choices in a more efficient way. Furthermore, we are not interested in the exact value of all levels, we are interested in the relative value of the utilities of each level with respect to the others. This poses an important efficiency increase as we only look for the best alternative rather than analyzing the whole space of possibilities. Notice that an important feature of the current research is that we allow to estimate categorical variables and main effects.

This research starts by analyzing how optimal designs have to be generated in order to conduct next CBC studies. A novel combination of traditional techniques with modern methods is explained i.e. Fedorov Algorithm. Then how parameter estimation is performed using Multinomial Logit Model. An important part of this Thesis is focused on understanding how to remove uninformative levels from the design in an iterative way. Finally, an extensive evaluation and application with simulated and real datasets is conducted to inspect how well the approach works.

Although at first sight another technique such as Adaptive Choice Based Conjoint Analysis (ACBC) may look similar, it is a completely different method from the one here proposed. Generally, ACBC tries to unveil part-worth utilities by generating a design with alternatives that slightly deviate from a user-defined most preferred alternative. This implies that the respondent has to define and create his preferred alternative beforehand. Then the space of alternatives is immediately reduced to those alternatives that are similar, with small deviations, from the user-defined choice.

The main goal of ACBC is to retrieve part-worth utilities to understand what features are important. In contrast, this research tries to reduce the whole space of possibilities iteratively until we get the best, most preferred, alternative. The approach here proposed aims to fill in the gap in the literature on CBC and propose a novel algorithm for CBC On-The-Fly (CBC-OTF).

This novel algorithm is intended to set the basis for further research in this direction as well as to serve as a practical tool for SKIM. The use of such an algorithm represents a step forward in Conjoint Analysis and an innovative technique in the Market Research field.

2 Literature Review

Conjoint Analysis is a very popular market research technique which has been used for more than forty years. It was introduced by Green and Rao (1971) and Green and Wind (1975) and they set the foundations of the field. They proposed a method to obtain valuations of the different characteristics of a product by asking people to rate them. Some developments on how to conduct conjoint analyses were done some years later by Green and Srinivasan (1978). Later contributions like those proposed by Louviere (1988) and Green and Srinivasan (1990) broadened the scope of the techniques, for instance by introducing CBC which aims to unveil part-worth utilities by asking respondents to select a product among several.

The need to develop faster and more efficient algorithms to design experiments was soon clear. The size of a full factorial design increases exponentially when increasing either the amount of attributes or levels. Several methods on Design of Experiments (DoE) were presented since the 70's on. They all have in common that the design is optimized with respect to some criterion. A design is said to be efficient when the parameters of the choice model are estimated with maximum precision (Zwerina et al., 1996). The natural approach is to generate an orthogonal design (Green and Rao, 1971) because of the extremely efficiency levels that it manages to achieve, however this is not always possible, for that reason we may seek other optimal approaches.

The most used optimal design is the $D-$Optimal design which tries to minimize the inverse of the OLS information matrix. Another approach is to use $G-$Optimality, where G is the criterion that minimizes the maximum variance in the experimental region (Wheeler, 2004). $I-$Optimality tries to minimize the average prediction variance. $A-$Optimal design seeks to minimize the trace of the variance-covariance matrix. $V-$Optimality minimizes the average prediction variance in the design space. Although they optimize with respect to different criteria these approaches have common characteristics. For instance when considering an OLS model, Lucas (1974) suggests that the best design in terms of $D-$Efficiency coincides with the best design in terms of $G-$Efficiency. Similarly Donev and Atkinson (1988) developed that $D-$Efficient designs are usually $G-$ and $V-$Efficient. In addition, Chasalow (1992) discovered that designs that are $A-$Optimal have usually a poor performance with respect to $D-$Optimality. All these Optimality designs are discussed more in detail in Huber and Zwerina (1996) and Reliasoft Corporation (2015).

$D-$Efficient choice designs are characterized by four different properties, as explained in Huber and Zwerina (1996). The first condition is *orthogonality* meaning that levels within attributes have to be independent from each other. The second one is *level balance*, each level appears with same frequency. *Minimal overlap*, the third property, which states that each alternative within a choice set does not have overlapping attribute levels. The fourth condition is *utility balance*, in the sense that within each choice set the utilities of alternatives are the same. In addition, the third property only makes sense for choice designs as the contrasts between attribute levels only are only meaningful compared to others within a choice set.

Having these conditions in mind there are several algorithms to achieve $D-$Optimality when dealing only with categorical variables. The most spread algorithm among marketers is an exchange algorithm developed by Fedorov (1972). This algorithm starts with a full factorial design and exchanges points with a candidate list. In every iteration the model computes the determinant of the information matrix. After using several random starts eventually it converges to a candidate global minimum. This procedure is quite slow. For this reason Atkinson and Donev (1989) consider a modified exchange algorithm, so-called BLKL, where only those candidate points with the largest prediction variance are exchanged for those with the smallest prediction variance of the design. Another exchange algorithm to achieve a $D-$Optimal is that one proposed by Meyer and Nachtsheim (1995), which only exchanges one factor level at a time. This leads to very low computational costs. Other valid approaches to generate D-Efficient designs but less popular are those proposed by Wynn (1972) where the added and deleted point are determined in different steps. Näther (1985) proposed an extension of this model to any kind of variance-covariance matrix. Mitchell (1974) proposed the DETMAX algorithm which makes the size of the design vary during the search procedure. Finally, Sandor and Wedel (2005) develop more extensively in the generation of heterogeneous designs.

However, Cook and Nachtsheim (1980), Galil and Kiefer (1980), Johnson and Nachtsheim (1983) and Nguyen and Miller (1992) arrive to the same conclusion: Fedorov algorithm requires more computation time than the other algorithms, but it manages to produce better designs.

Once the design is generated and the survey is distributed to respondents one has to compute part-worth utilities. This is done by the classical approach of Multinomial Logit Model proposed by Cox (1958) and Walker and Duncan (1967). A general overview of choice modeling on the Conjoint Analysis field is discussed in Chrzan and Orme (2000). In order to improve the accuracy and stability of parameter estimates a common approach is to combine part-worth utilities with Hierarchical Bayes Halme and Kallio (2011). In addition, this procedure allows for heterogeneity among respondents. For this purpose the use of MCMC techniques like Gibbs sampling Casella and George (1992) and Thompson (1933) become necessary. More developed Bayesian concepts can be found in the book written by Gelman et al. (2014). However, less computational intensive techniques such as mixed logit might be considered in order to speed up the algorithms.

Finally, in order to create an algorithm for design generation on-the-fly it is necessary to use criteria to assess which levels should be removed in each iteration so as to obtain a new more insightful design. For that reason we need to compare between probability distributions. A widespread and useful measure to assess if two distributions are similar is the Kolmogorov-Smirnov Test Massey Jr (1951) and Lilliefors (1967). Other authors like Wilcox et al. (2014) develop different methods to compare independent groups via lower and upper quantiles. Bayesian methods for the comparison of distributions are also interesting approaches like those proposed by Borgwardt and Ghahramani (2009) and Betancourt (2010). This research focuses on comparing distributions by integrating the common area shared.

Notice that this research represents a new line of research in Choice-Based Conjoint Analysis. A relatively new and similar method to that proposed here is Adapative Choice-Based Conjoint Analysis which departs from a user-defined best alternative, as explained in detail by Huber et al. (2003). Although there are similarities like the optimization of tasks, there are huge differences: in ACBC, the user-defined task reduces the space of possibilities to its neighbors meaning that the initial choice has large influence in subsequent tasks Toubia et al. (2004). Hence other methods for obtaining an Optimal design are used. Only very few attributes, 2 or 3, vary in every task and every certain tasks the respondent has to answer if a certain level is unacceptable. Although this caveats, ACBC usually leads to better results compared to traditional CBC by accepting the fact that results may be biased to some extent by the initial choice Green et al. (1991).

3 Choice Based Conjoint Analysis

Each and every day we face the dilemma of which product to choose in the marketplace. Some decisions are trivial and easy to do, for instance if we want a white or black chocolate bar. Others involve a thorougher evaluation of the possible trade-offs as they are complex, because of the amount of features or by the inherent complexity of the product, for example an insurance plan. However, in both examples a decision making process is involved, which usually leads to a comparison between competing alternatives. Consumers make their decisions based on their individual preferences which may be or may not be aligned with overall market preferences. But what features have more importance in the process of evaluation? What trade-offs between attributes can be done? How can firms maximize their profit by selecting the appropriate characteristics?

Conjoint Analysis is an econometric approach which aims to determine the value of each of the different attributes that form a product or service. The main objective of CBC is to obtain part-worth estimates for all attribute levels, in order to know which ones are more important. To carry out such an analysis we need to conduct surveys where respondents face different product alternatives, so implicitly they have to evaluate trade-offs between products. A fundamental premise when analyzing the importance of different attributes, is that people find it difficult to evaluate each feature independently. Using a more realistic setting by confronting respondents to choose between different realistic choice situations we are able to disentangle the importance of each feature.

As an illustrative example for mobile phones we consider four different attributes, features, with different levels for each attribute: *size*, *RAM*, *screen*, *connection* as in Table 1. On each task the respondent is required to select one of the different alternatives, also called concepts, presented. By repeating this process several times, for instance six different choice tasks, we are able to estimate utility part-worths and disentangle what features are most important.

In Table 1 one can see an example of a Choice Based Conjoint Analysis task of technological devices. The respondent has to select the most preferred alternative among the four displayed. In this case he chooses Concept 2 as the most preferred one.

Table 1: Example of different mobile phones in a choice task in CBC.

	Concept 1	Concept 2	Concept 3	Concept 4
Size	8'	12'	10'	14'
RAM	1GB	4GB	2GB	6GB
Screen	4K	HD	HD	4K
Connection	3G	4G	2G	4G
	☐	✓	☐	☐

An interesting fact of CBC is that there is no need to evaluate all possible profiles to derive utilities. First, because it is too costly and in most of the cases impossible, e.g. asking thousands of combinations to a single respondent would be unfeasible. By making the right tasks, those that maximize information thus confront with difficult trade-offs, we are able to attain equally good estimates.

In a typical CBC study we have the following elements:

- **Choice task**: Selection of alternatives where respondents have to choose the most preferred one.
- **Profile/Concept**: Each of the different alternatives presented in a choice task.
- **Attributes**: List of features that conform the product.
- **Levels**: Each of the different possibilities that an attribute can take.

3.1 Illustrative example

Recall that the objective of this research is to find the very best combination of attribute levels. We consider the following example to better understand how this is achieved. A pharmaceutical company wants to introduce in the market a new drug to treat fleas on cats. What is the best product design to attract the largest amount of customers?

Figure 1 depicts the different features the package has and the different characteristics, levels, that they can take. In Table 2 there is a more explicit description of each attribute and its levels e.g. pictures, claims, colors...

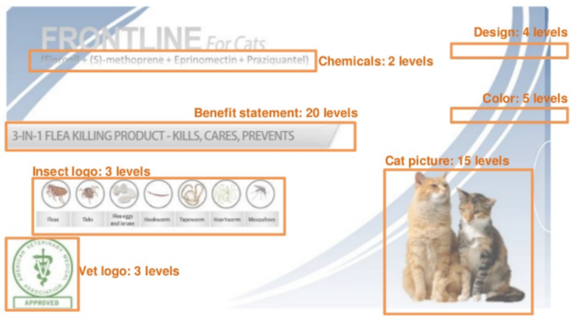

Figure 1: There are 7 different features, attributes, with different characteristics, levels. The total amount of profiles is 108,000.

Table 2: There are 108,000 possible combinations. Testing them all is unfeasible.

	Attributes	Levels	Nature
1	Package Concept	4	Visual Concepts
2	Pet Picture	15	Pet pictures
3	Package Color	5	Integrated in Pack Concepts
4	Claims	20	Text Statements
5	List of Chemical Components	2	Present/Absent
6	Icons	3	Present (2 versions)/Absent
7	Vet only icon	3	Present (2 versions)/Absent
	# of possible combinations	108,000	

Given that there are 108,000 unique alternatives we cannot test them all because of the large costs in terms of time and money associated. Furthermore it would be too inefficient, and most likely unfeasible, to ask for all the combinations. Notice that this research only considers categorical variables as shown in the example in Table 1. Traditional CBC would form questions based on the most informative alternatives, according to a criteria e.g. D-Efficiency, and then ask to n respondents to select the preferred one in different choice tasks. As stated previously there is no actual learning during the whole process, we only learn once about all the features and all individuals. Furthermore, learning about bad features is not interesting. This research overcomes this issue by iteratively removing the most uninformative alternatives and conducting a CBC analysis iteratively up to the point that a candidate global optimum is reached. That is the alternative that maximizes the utility to most of the people. For illustrative purposes, we could consider Figure 2 as the most preferred alternative extracted from the OTF algorithm.

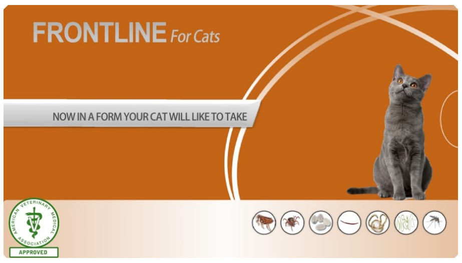

Figure 2: After iteratively removing the least informative levels and conducting CBC analyses iteratively we find that this is the best alternative, so the package that the company should use for the product.

4 Methods

This section develops the main algorithm. It starts from the basics, what a design is, how to generate optimal designs and it builds up the whole algorithm: estimation, level assessment and removal. The objective is to give clear insights on how the algorithm works and operates for further developers and researchers who may be interested in implementations in other languages such as $C++$, Java etc. or practitioners who might be interested in the application of this algorithm and want more technical details.

Recall that the main objective of this algorithm is to obtain a global optimum, that is the best combination of levels and attributes that make the product preferred by everyone. We can consider two types of optimality: individual and population. The first analyzes how optimal the final alternative is with respect to individual utilities. The latter seeks a global optimum at population level, that is the alternative that maximizes the utility to a great majority of individuals. The structure of this section follows the same order as Algorithm 1, which states how the OTF algorithm works in pseudo-code.

The idea behind this algorithm comes after careful consideration of how we could achieve individual and global optimality in iterative CBC studies. Previous internal research at SKIM conducted mainly by K. Van der Wagt, SKIM Research Director, set the foundations of how this framework could be stated and developed given their extensive expertise and market knowledge. However, there was nothing at all implemented nor developed formally in addition that many problems had to be sort out.

The algorithm first seeks to create an optimal design regarding some criteria so as to make the most efficient questionnaires possible. Next, we conduct a CBC study to different individuals and compute the population utility of each level. Finally, by using a Pseudo-Density Function and probability functions we remove one level per iteration. In case that one attribute has too many levels, one level from this feature will also be removed in order to avoid an undesired effect called attribute importance. Once one or two levels have been removed, we will proceed with generating a new D-Optimal design with a novel technique here presented. This whole procedure is repeated the desired number of times until a candidate global optimum is reached.

```
Data: Full Factorial Design
Result: Most preferred product
for $m$ iterations do
    Generate a D-Optimal design using Fedorov Algorithm;
    Ask $n$ respondents;
    Compute utility part-worths with Logistic Regression based on last $n$ respondents;
    Perform assessment tests of the levels;
    Remove uninformative levels;
end
```

Algorithm 1: Design Generation On-the-Fly Algorithm

This section starts from the basics by explaining what Choice Based Conjoint Analysis is. Then we move to design generation and a novel approach based on Fedorov Algorithm proposed in this research. Then the use of multinomial logistic regression for parameter estimation is covered. We proceed by explaining how the model evaluates and decides which the *worst* level is. Finally a performance evaluation function is provided.

4.1 Design

In Choice Based Conjoint Analysis, a design is defined as a the total sum of tasks across all respondents. It is paramount to generate the most efficient design possible according to some criteria. The more efficient a design is, the more precise parameter estimates are.

The first step is to generate a full factorial design, that is a design with all possible alternatives. This full factorial design usually tends to be very large and the size increases exponentially with the number of attributes and levels. If all attributes have the same amount of levels the size of the design is defined as k^l, where k are the attributes and l the levels. Given that showing all possible alternatives is not feasible nor efficient we seek to generate a smaller, but more efficient design which can be administered to n respondents.

One of the most used criteria to generate optimal designs is the D−Efficiency criterion. We seek that specific combination of n_D unique alternatives that minimize the determinant of the inverse of the information matrix. The size n_D can be defined by the user so there is more room for customization, in this research we use n_D to be a 75% of the full factorial design so the speed and efficiency are increased compared to lower proportions. Note that the Information matrix is defined as the negative expected value of the Hessian matrix. The Hessian Matrix is the second derivative of the Likelihood Function. The general expression is specified as

$$I(\hat{\beta}) = -E\left[\frac{\partial^2 l}{\partial\hat{\beta}_i \partial\hat{\beta}_j}\right]. \tag{4.1}$$

Notice that in this research although we make use of logit type models we use the general OLS Information matrix expression as criterion to assess the efficiency of the design. The use of the OLS information matrix is done in order to speed up computations as both methods usually lead to very similar results, as shown in Wheeler (2004) and Rao et al. (2014).

So as to obtain the highest D-Efficiency we need to minimize the determinant of the inverse of the OLS information matrix. To do so we define X as the matrix coding the design such that

$$\arg\min |(X'X)^{-1}|. \tag{4.2}$$

There are several methods to generate D-Optimal designs. Most softwares packages rely on randomization algorithms for design generation (Johnson et al., 2013). This is the case for Sawtooth Software (SSI), the most spread tool among market research agencies, which only allows to generate random designs but using four slightly different criteria. These methods are:

- Complete Enumeration: Seeks to generate a nearly orthogonal design in terms of main effects. Within choice sets, levels are duplicated as little as possible.
- Shortcut: Each profile is built based on the selection of the least often previously used level for each specific respondent.
- Random: Profiles are selected completely at random.
- Balanced Overlap: This approach lies between Random and Complete Enumeration. It uses the same logic as the latter but it allows for overlap to some extent.

A usual approach among practitioners, as done at SKIM, is the use of Balanced Overlap as mechanism to generate designs. Interestingly, the software SSI reports the D-Efficiency but it does not seek to maximize such criterion. A more detailed explanation on these methods can be found on Chrzan and Orme (2000).

The approach proposed in this research relies heavily on the generation of a highly optimal design in terms of D-efficiency. This approach allows us to obtain more reliable utility estimations. To do so, we use the Fedorov Algorithm as proposed by Fedorov (1972). This method tries to maximize the information matrix by starting with a random design with a user-defined m unique alternatives. Then it creates a candidate list of random alternatives from the full factorial design. Then the algorithm exchanges the points of the candidate list with those of the covariance matrix. If the determinant of the inverse of the information matrix is smaller then it keeps the new alternative. In mathematical notation it works as follows: first consider a $k \times k$ variance-covariance matrix M. The update of M involves removing the k vector y and then replacing it with the k vector x. This specified as $M + FF'$, where $F = (x, iy)$ and $i^2 = -1$ using imaginary numbers. By applying standard algebra rules we obtain

$$\begin{aligned} M_1^{-1} &= (M + FF')^{-1}, \\ M_1^{-1} &= M^{-1} - M^{-1}F(I_2 + F'M^{-1}F)^{-1}F'M^{-1}, \\ |M_1| &= |M + FF'| = |M| \cdot |I_2 + F'M^{-1}F|. \end{aligned} \tag{4.3}$$

If we define $d_{uv} = u'M^{-1}v$, and $d_u \equiv d_{uu}$, then $|I_2 + F'M^{-1}F| = (1 + d_x)(1 - d_y) + d_{xy}^2 = 1 + [d_x - (d_x d_y - d_{xy}^2) - d_y] = 1 + \Delta_D$. $max\Delta_D$ is chosen among the candidate vectors to add in the algorithm. In other words, we include those vectors that maximize the available information.

An extension of this Algorithm is the so-called Modified Fedorov Algorithm or BLKL Algorithm, proposed by Atkinson and Donev (1989). This method works as the original Fedorov Algorithm but it only exchanges those points with minimum variance from the dispersion matrix with those with maximum variance from the candidate list. This approach speeds up the process but it usually leads to slightly worse D-efficient designs as the exchanged points are fewer.

Because the Fedorov Algorithm is a highly dependent on the initial random design and candidate list the optimization procedure is performed several times until convergence to a global maximum. Fedorov (1972) suggests that 10 iterations maximum are enough due to marginal increase in efficiency and high computational costs. However, notice that this approach was proposed in the early 70's when computational costs were extremely high compared to today. For this reason in this paper we make use of more than 35 initial random starts to ensure we do not get stuck in a local optimum.

Recall that this whole procedure of generating a D-optimal design is only applied on the first iteration of the algorithm. The main objective is to have a fast method, so generating a completely new D-optimal design on every iteration would be better in terms of design efficiency, but not in practical terms due to its high computational requirements. For that reason, in the subsequent iterations what it is done is excluding from the design those alternatives that include the removed level. Then, we seek those alternatives that do not include the removed level that maximize the D-Efficiency criteria in Equation 4.2. In short, old solution becomes the starting point for the next iteration.

Once the optimized design has been created we need to construct different choice sets. Following the standard approach commonly applied by most important software packages such as SSI, we randomize the alternatives and administer them in the desired size. For example, we may want 6 different choice tasks with 4 different concepts per choice task. This selection can be made by the researcher. However, a usual setting is to use between 4 to 6 choice tasks and 3 to 5 concepts. This is justified by the fact that if there are too many tasks the respondent may get tired and fill in the survey randomly. In addition, if there are too many concepts per choice task then the evaluation of trade-offs may be too large and difficult thus respondents may reduce the decision process to just evaluating very few features.

An example of the Fedorov Algorithm when trying to re-optimize the design can be found in Tables 3 and 4. We can see that instead of generating a completely new design, we opt for just substituting those alternatives which included the undesired level.

Table 3: Suppose we have 3 attributes with 3 levels each, and we generate the following D-Optimal design with 9 different alternatives. In this example we want to remove Level 2 from Attribute 2. In bold you can see those alternatives that include this undesired level.

	Attribute 1	Attribute 2	Attribute 3
Alternative 1	1	1	1
Alternative 2	2	1	1
Alternative 3	3	1	1
Alternative 4	3	1	2
Alternative 5	**2**	**2**	**2**
Alternative 6	3	1	3
Alternative 7	2	3	2
Alternative 8	**1**	**2**	**2**
Alternative 9	1	3	3

After exchanging only those alternatives that contain the second level of the second attribute, we end up with a highly efficient design with very low computational costs. In Table 3 one can see how the original design is, and in Table 4 how it would look like after removing one level.

Table 4: We only exchange Alternative 5 and 8 to generate a new D-Optimal design. We achieve high efficiency with very low computational costs.

	Attribute 1	Attribute 2	Attribute 3
Alternative 1	1	1	1
Alternative 2	2	1	1
Alternative 3	3	1	1
Alternative 4	3	1	2
Alternative 5	**1**	**3**	**2**
Alternative 6	3	1	3
Alternative 7	2	3	2
Alternative 8	**1**	**1**	**3**
Alternative 9	1	3	3

Notice that rarely, and just by chance, we might find a singular design. That implies that there are linear dependencies among variables. In classical statistics this is usually referred as collinearity. In such a case we will generate a completely new D-Optimal design considering only the levels that were not removed. Although a singular design leads to infinite determinants of the inverse matrix thus it does not represent a problem from a theoretical standpoint, from the practical point of view might lead to technical problems when using software and existing packages e.g. R libraries. However, this is not a serious problem as it is an extremely infrequent situation.

4.2 Utilities Simulation

This paper proposes a theoretical approach to the algorithm and models used. It uses simulated data so as to have full control of the utilities thus evaluate the performance at individual and population level with respect to standard approaches like CBC. The natural next step, which is left for further research, is a real life implementation with real respondents.

We generate a matrix of $n \times \kappa$ utilities, where n is the number of respondents and κ the total number of levels in the design, so each individual has a utility for each level of the design. We can then draw utilities from different distributions with certain means and covariance matrix, so we can have full control of which levels are more desired than others. A standard approach used in this research consists of drawing from a Multivariate Normal distribution such that

$$\begin{aligned} &U_i \sim \mathcal{N}(\mu, \Sigma), \\ &\mu = [\mu_1, \mu_2, \dots, \mu_\kappa], \\ &\Sigma = \begin{bmatrix} \sigma_{11}^2 & \dots & \sigma_{1\kappa} \\ \vdots & \ddots & \vdots \\ \sigma_{\kappa 1} & \dots & \sigma_{\kappa\kappa}^2 \end{bmatrix}. \end{aligned} \tag{4.4}$$

Next, we compute the utility of each of the alternatives presented to each respondent on every task. This is done by encoding the design in a dummy format and using the true simulated utilities to obtain an alternative specific utility.

4.3 Dependent Variable Generation

One of the issues that one has to sort out when working with simulated data is the fact that we need to generate a dependent variable, 0/1, whether the concept c out of C concepts was chosen or not. This allows us to have a response encoded in a binary format, as in a standard CBC, which eases the process of estimation in later stages of the modeling framework.

Firstly, we add to the simulated utility a random number out of a Generalized Extreme Value distribution Type-I so as to modify the utilities. We specify a Gumbel Error with $\mu = 0$ and scale parameter $\beta = 1$. The procedure is defined as

$$\tilde{u}_{ict} = u_{ict} + \varepsilon_{itc} \text{ where } \varepsilon_{itc} \sim GEV(0,1) \tag{4.5}$$

Then we select that concept with highest final utility such that

$$\arg\max Pr(Y = 1|\tilde{U_{it}}) = max\{\tilde{u}_{i1t}, \tilde{u}_{i2t}, \dots, \tilde{u}_{itc}\} \tag{4.6}$$

An example can be found in Table 5. We have four different concepts with their simulated utilities and we add them a Gumbel Error so we introduce some noise in the utilities.

Table 5: We have four different concepts with their simulated utilities. By adding a Gumbel Error to them we obtain new utilities. The chosen is that concept with highest utility, in this example Concept 4.

	u_{itc}	Error	$\tilde{u}_{itc}$	Chosen
u_{it1}	2.5	-0.7	1.8	0
u_{it2}	1	0.4	1.4	0
u_{it3}	1.6	-0.3	1.3	0
u_{it4}	2	1.1	3.1	1

4.4 Estimation

Two usual approaches in CBC studies consist of estimating the model parameters with Bayesian Methods or Aggregated Logit models. The first approach allows to estimate individual utilities thus obtaining a density function for each of the levels in the study. The later obtains utilities at population means, hence a single parameter per level.

A research carried by Huber and Train (2001) tried to shed light on the performance of Bayesian methods compared to Maximum Likelihood methods, like logistic regression. They found almost identical results regarding the utility part-worths for the attributes. They also found that prediction of a holdout task was almost identical. They concluded that Bayesian techniques are useful when there is a large number of part-worths to estimate, because of the possible existence of local maximum given certain distributions. They argue that in such a case, a Bayesian method is better also because identification of parameters is not a problem due to the use of prior information. For further information see Huber and Train (2001). The main drawback of Bayesian techniques are the high computational costs and the large amount of time needed for convergence of the estimates. For these reasons, in this research an aggregated multinomial logit model is preferred due to its high computation speed and also reliable and precise estimates.

The general expression for a multinomial logit model is defined as

$$Pr(Y_{ij} = j) = \frac{\exp(\beta_j X_i)}{\sum\limits_{k=1}^{J-1} \exp(\beta_j X_i)} \tag{4.7}$$

Further information on the Maximum Likelihood Estimation for multinomial logit models can be found in Appendix A.

Using this technique we manage to obtain utility estimates for every level. Notice that because the variables used are categories, we need to encode it in dummy format so as to estimate the parameters. In addition, every attribute considers that there is a base or reference category, that means that $\beta_i = 0$, thus utility of concept i is assumed to be 0. This implies that all parameter estimates within the same attribute are be compared with respect to the reference group.

4.5 Level Assessment

This subsection is divided in three parts: first of all we study what the Pseudo-Density Function presented is and how it is constructed. In the second part we analyze how the candidate levels to be removed are compared by integrating functions. Finally, we develop how the levels are removed by using probability functions.

4.5.1 Pseudo-Density Function

A major drawback in the estimation is that every attribute needs to have a reference group which assumes $\beta_{jl} = 0$, in turn implying that there is no standard error for the base category.

To overcome such issue we assume that the utility density of each level is normally distributed, $f(\hat{\beta}_{jl}) = N(\beta, s_{jl})$ where jl corresponds to the l level of the j attribute. From this we can obtain a probability density function for every estimated level except the base category of each attribute. Note that the reference category is defined as l^*.

We introduce the concept of Pseudo-Density Function as the probability density function of the utility of each level including the reference group. In order to make the levels comparable we assume $s_{jl^*} = \frac{1}{L} \sum\limits_{c=1}^{C} s_{jl}$ for l^*.

$$f(\beta_{jl}) = \begin{cases} N(\hat{\beta}, s_{jl}), & \text{if } l \neq l^* \\ N(0, \frac{1}{n} \sum\limits_{l=1}^{L} s_{jl}), & \text{if } l = l^* \end{cases} \tag{4.8}$$

This is a very fast and efficient approach which proves to be useful for the problem in hand. From this it is clear the name of Pseudo-Density Function, because we assume a certain standard error for the base level.

4.5.2 Level comparison

Once all the *p.d.f.* for every level of each attribute are obtained we proceed by selecting the two levels with smallest $\hat{\beta}$ of each attribute. This implies that we select the levels with smallest utility on average.

$$S_j = \min_2\{\hat{\beta}_{J1}, \ldots, \hat{\beta}_{jL}\} \tag{4.9}$$

Then the $\hat{\beta}$'s contained in every set S_j are compared. To do so we integrate the area between the two regions that both utility *p.d.f.* cover. We use their corresponding $\hat{\beta}$ with their respective standard error $\hat{s}$ estimate. Recall that for the reference category of each attribute we assumed $\hat{\beta}_{jl^*} = 0$ and $\hat{s}_{jl} = \frac{1}{n}\sum_{l=1}^{L} s_{jl}$.

We decide to select the two smallest levels because the idea of the framework is to remove one level per iteration. To do so we only want to compare few distributions so as to the speed the whole iterative process. In addition, by comparing only two distributions at a time we can integrate the shared area. If the area is very small it means that they are very separated distributions. So the distance between the worst and the second worst is large.

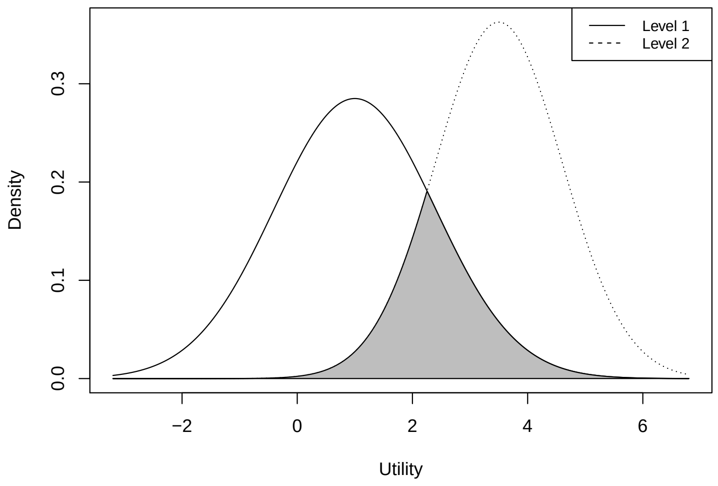

Figure 3: We compute the integral of the gray area in order to assess which levels are worse. Level 1 corresponds to the worst level and Level 2 to the second worst level of attribute j in terms of average utility.

We compute the integral for each of the two worst levels per attribute. For instance, if we have six attributes we will end up with 6 different plots like in Figure 3 comparing the two worst levels of each attribute, namely l_1 and l_2. The integration of this shared area gives the total mass that both distributions have in common.

In formal notation we define

- $X_1 \sim N(\mu_1, \sigma_1^2)$ with p.d.f $f_1(x_1)$ and CDF $F_1(x_1)$
- $X_2 \sim N(\mu_2, \sigma_2^2)$ with p.d.f $f_2(x_2)$ and CDF $F_2(x_2)$

If we assume $\mu_1 < \mu_2$ and denote with c the point of intersection where both probability density functions meet, then we can compute the gray area, the intersected part as:

$$P(X_1 > c) + P(X_2 < c) = 1 - F_1(c) + F_2(c) = 1 - \frac{1}{2}erf\left(\frac{c-\mu_1}{\sqrt{2}\sigma_1}\right) + \frac{1}{2}erf\left(\frac{c-\mu_2}{\sqrt{2}\sigma_2}\right) \tag{4.10}$$

where erf is the Gauss Error Function.

4.5.3 Level Removal

Once we have selected the attribute whose two *worst* levels have smallest overlap, we need to decide which of the two levels we remove. For this purpose we generate a probability function so as to generalize the model. The reason behind this decision is that in some cases we might find two distributions highly overlapped. In such a case we are not sure which one is really worse than the other, so a probability function to decide which one to remove helps to generalize the model and take into account such situations. Then, with probability π we will remove Level 1 and with probability $1 - \pi$ we will remove the second level. The algorithm allows the user to use two slightly different versions of the probability functions, depending whether she wants to take into account the variance of the estimator or not. The general approach to generate a valid probability function that assigns higher probability to smaller values. To do so we need a transformation denoted as t of random numbers A and B: $P(A) = \frac{t(A)}{t(A)+t(B)}$. Where t is positive and decreasing such that smaller values obtain higher probability and to ensure that probability is defined between 0 and 1.

In this research we consider $t(x) = \log(1/\Phi(\frac{x}{\omega}))$, where $\omega > 0$ and Φ is the standard normal CDF. The default setting of the algorithm here developed is set at $\omega = 5$. In fact, we could use any monotonic decreasing survivor function as transformation and we would obtain similar results.

If we only consider the parameter estimate $\hat{\beta}_{jl}$ then the probability of removing level jl_1 is

$$\pi_{jl_1} = \frac{\log\left(\frac{1}{\Phi(\frac{\hat{\beta}_{jl_1}}{\omega})}\right)}{\sum_{i=1}^{2} \log\left(\frac{1}{\Phi(\frac{\hat{\beta}_{jl_i}}{\omega})}\right)} \tag{4.11}$$

This mapping function comes to say that the more negative $\hat{\beta}_{jl}$ is the higher probability to be removed. This is in line with common sense and statistical theory: the smaller the utility of a level the worse compared to the other parameters.

In case we wanted to take into account the variance of the distribution we can use the standard error of the parameter estimate as normalizing factor, which is equivalent as using the t-value of a parameter, such that

$$\pi_{jl_1} = \frac{log(\frac{1}{\Phi(\frac{\hat{\beta}_{jl_1}/s_{jl}}{\omega})})}{\sum_{i=1}^{2} log(\frac{1}{\Phi(\frac{\hat{\beta}_{jl_i}/s_{jl_i}}{\omega})})}. \tag{4.12}$$

By applying this probability function we will obtain two different probabilities, π and $1 - \pi$, one for each of the two candidate levels to be removed. So the next step is to remove level l_1 of attribute j with probability π.

The main objective is to remove one level per iteration, nonetheless we might find important drawbacks if we keep an attribute with too many levels. The levels of this attribute will be probably very close to each other and they may not be removed until later iterations. The fact that an attribute with too many levels biases results in different ways is known as attribute importance and has been very studied in Green and Srinivasan (1978), Wittink et al. (1990) and Orme (2002), especially in the field of human behavior biases.

In order to solve this problem regarding attribute importance we also apply the same algorithmic procedure to that attribute which has too many levels. In order to assess what "too many" is, we use the following rule of thumb: If an attribute has more levels than the average levels plus a standard deviation we will also remove that level from the design. We can formalize this statement as following:

Let $v = \{v_1, v_2, \ldots, v_{j-1}, v_j\}$ be a vector of size j where each element corresponds to the number of levels each attribute j has. Denote with S_j a vector which contains, in a dummy encoding, the attribute from which a level will be removed, i.e. 0 we do not remove, 1 if we do.

$$S_j = \begin{cases} 1, & \text{if } v_j \geq \frac{1}{j}\sum_{i=1}^{j} v_i + \hat{\sigma}_j \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

For instance, a $v = \{3, 6, 2, 10\}$ would imply four attributes with three, six, two and ten levels respectively. The average number of attributes is 5.25 with a standard deviation of 3.59. This implies that those attributes with more than 8.84 levels, rounding to the closest integer 9, will have a level removed. In this particular example we would have $S = \{0, 0, 0, 1\}$ so a part from removing the *worst* level overall, we would also remove those levels where $S_j = 1$.

4.6 Link with probabilities

The approach here presented relies heavily on analytical integration of distribution of parameters. Another possible implementation of this algorithm could consider and work with probabilities out of the parameter estimates. As previously explained we generalize the distribution of the coefficients, including the reference group, assuming a normal distribution and a standard error for the base category. This method is backed-up by statistical theory concerning multinomial logistic regression.

Once we have run our multinomial logistic regression we obtain parameter estimates for all the alternatives except to a reference group. Having j different attributes implies having j reference groups as well, one per attribute.

Recall the specification of a basic multinomial logit model with one single regressor

$$\begin{aligned} \text{logit}(\pi_{jl}) &= \log\frac{\pi_{jl}}{\pi_{Jl}} = \beta_j + \beta_{jl}X_{jl} \\ \pi_{jl} &= \frac{\exp(\beta_0 + \beta_{j1}X_{jl})}{\sum_{j=1}^{J}\exp(\beta_0 + \beta_{jl}X_{jl})} \end{aligned} \tag{4.14}$$

We need to transform to odds Ratio by exponentiating the coefficient such that $\exp(\hat{\beta}_{jl})$. As a reminder, in general terms, the odds ratio represent the constant effect of a predictor X on the likelihood that one outcome will occur. From odds ratio we can easily go to probabilities of the success, as presented in Equation 4.14.

Notice that one can easily derive the marginal effect, which is the change in the probability of choice of the kth brand with respect to changes in the $j - th$ x variable for one individual. When dealing with dummy variables, the effect of that variable on the probability is stated as the difference in the probabilities computed between that variable being 1 and 0. From this it is clear that the effect of any variable is not constant as it varies with the value of the probability of choice. For further details see Rao et al. (2014).

This explanation helps us to understand why our approach is a valid way to tackle the issue of selecting which levels to remove. In our approach we choose the worst two levels per attribute. These two levels have a direct connection with the odds ratio and probabilities associated. They turn out to be *the worst* ones in the sense that they are less likely to be chosen.

As an illustrative example we can consider one single attribute with 4 different levels. After running the algorithm and performing the multinomial logistic regression we obtain the following parameter estimates $\beta_j = \{0, 3, -1, 2\}$. From this we can already know the ordering: $\beta_3 < \beta_1 < \beta_4 < \beta_2$. We can also compute the odds ratio and probabilities as presented in Table 6.

A crucial conclusion emerges from Table 6, by using our algorithm the two candidate levels to be removed would be the the third or the first. If we used probabilities to decide which the candidate levels to be removed are, as in Table 6, we would obtain the same answer. This is because of the direct link between coefficient estimates and probabilities, which make using only the parameter estimates an equally efficient and faster approach than computing probabilities on each iteration. Given that we prefer high computational speed, we stick to the algorithm presented instead of computing odds and probabilities.

Table 6: Illustrative example of probabilities on Logistic Regression.

	Estimate	Odds	Probability
β_1	0	1	0.034
β_2	3	20.08	0.697
β_3	-1	0.36	0.013
β_4	2	7.38	0.256

4.7 Simulation Assessment

4.7.1 Individual Level

Once the candidate global optimum has been achieved, we need a measure to know how the model performs, that is if the final combination of levels is the most preferred alternative among all. We can use completely simulated individual utilities, for instance from a multivariate normal distribution, or we can also use individual utilities from previous studies carried out by SKIM. This approach allows us to have a better control and understanding what levels are preferred before even starting the algorithmic procedure presented in this paper.

The way to proceed consists of creating an individual scale from 0 to 1, called Z-Scale, where 0 is the worst possible combination, and 1 the best possible alternative, and then locating the final product from the algorithm in this scale. Doing such a procedure yields a standardized individual ranking of the final product with respect to the worst and best alternatives.

$$z_i^* = \frac{U_i^* - min(U_i)}{max(U_i) - min(U_i)} \tag{4.15}$$

where $min(U_i)$ corresponds to the utility of the worst combination of levels possible for individual i, $max(U_i)$ is the maximum utility possible, and U_i^* is the utility of the final product derived from the algorithm presented. This allows us to create a distribution of the position of the candidate global optimum achieved in a scale from 0 to 1 and see if we really end up far away from the individual optimum. We can also generate confidence intervals out of the distribution, for instance by taking the 5th and 95th quantile. This approach can be seen as an extension of the usual Hit Rate measure, which is the average of correctly predicted best choices over all respondents.

As a measure of comparison we use a multinomial logit model considering all levels and attributes and then selecting those levels with highest utility. This proofs useful as we can compare our algorithm with a standard CBC approach. By computing the Z-Values for our method versus a standard CBC approach we can understand if there is an improvement in performance in selecting the most preferred alternative.

4.7.2 Population Level

Another approach to evaluate the performance of the algorithm is to look at the optimality of the final alternative at population level. Given that we simulate the utilities we can compute the average true utilities per level. The levels with highest average utility will be considered as the true optimal product. This alternative is compared to the final alternative output by the algorithm. An example can bee seen in Table 7. We have four different attributes where the first column displays the best level and the second the worst level. The third column is the predicted level by the algorithm. The last one whether the prediction was correct or not.

Table 7: Example of optimality at population level. Three out of four attributes are correctly predicted.

	Best	Worst	Predicted	Correct
Attribute 1	5	3	5	✓
Attribute 2	6	10	9	
Attribute 3	2	4	2	✓
Attribute 4	3	2	3	✓

5 Data

In order to evaluate the performance of the algorithm we need to test it with data. This research uses two kinds of data, first of all we simulate individual utilities with certain means and covariance matrix. This allows us to have full control of the distributional assumptions of our real utilities, thus making possible all kinds of variations. In addition to the simulated data we use real individual utilities extracted from previous studies at SKIM. This proves useful to see if the algorithm converges to the candidate global optimum in real situations, which usually are trickier due to the fact that people's preferences may not follow standard distributions.

For the empirical part we use five different datasets from different studies which contained individual part-worth utilities retrieved from Hierarchical Bayes estimation. Due to intellectual property issues, most of the information on the attributes cannot be disclosed.

- The first dataset which contains utilities is from a CBC study for a Dutch telecommunications company. There are 8 different attributes with the following number of levels each one: $v_1 = \{6,3,7,5,2,5,5,6\}$. The study tried to unveil what factors make a phone rate more attractive. In total the dataset has 2057 different individuals with their respective utility-part worths.
- The following dataset used in this research concerns an important bank in the EU. It analyzes what features influence people the most when opening a bank account. It has 5 attributes with $v_2 = 7,3,3,4,4$ and 242 different individuals.
- The third dataset has 9 attributes with the following levels $v_3 = \{3,3,5,4,3,4,3,4,5\}$. As aforementioned, due to IP the utilities are unlabeled so it is impossible to know what it is about. We have 817 different individuals in total.
- The fourth dataset used counts with 6 attributes with levels $v_4 = \{8,17,4,5,4,3\}$, and 1723 different individuals. It is analysis about the banking sector, but no further information can be provided.
- Finally, the fifth empirical data with its corresponding individual utilities has 8 attributes and levels $v_5 = \{12,3,2,3,2,3,5,6\}$ and a total of 273 individuals. It is a CBC study for the automotive industry.

As one can see the different datasets used come from different industries, with different numbers of attributes, levels and individuals. This is very helpful as we can compare the behavior of the algorithm under different circumstances and scenarios. Recall that we only measure main effects, so possible interaction between attributes and levels are not taken into account. On the other hand, given that data is unlabeled we cannot easily distinguish interactions. For instance, the variable *price* usually has interaction with other terms. As we do not know which variable is price this cannot be taken into account. The fact of only computing main effects may lead to some bias in the estimates of the model thus to sub-optimal results. However, we leave this for further research.

6 Results

6.1 Design Generation

A central pillar of the present research is the capability of generating new D-Optimal designs in a very short time. Using such procedure poses an improvement with respect to traditional software like SSI: the efficiency of the design is higher than using a random sampling of alternatives trying to keep the levels balanced, and computation time is lower. For instance, in Figure 4 we generate a design with 1,000 alternatives and 7 attributes whose number of levels are $v = \{3, 3, 3, 3, 3, 3, 3\}$, a 3^7 design, takes only 0.01 seconds. However, as we increase the number of levels per attribute, going from a 4^7 to a 8^7 design, we observe that the computation time increases exponentially up to the point that generating a 8^7 design takes around 5.7 hours. This can be explained by the fact that the design is extremely large, there are 2,097,152 unique alternatives, and from these ones we have to select only 1,000. This process becomes very slow when the full factorial design is very large. Nonetheless, studies with so many attributes are not common in practice due to its practical limitations.

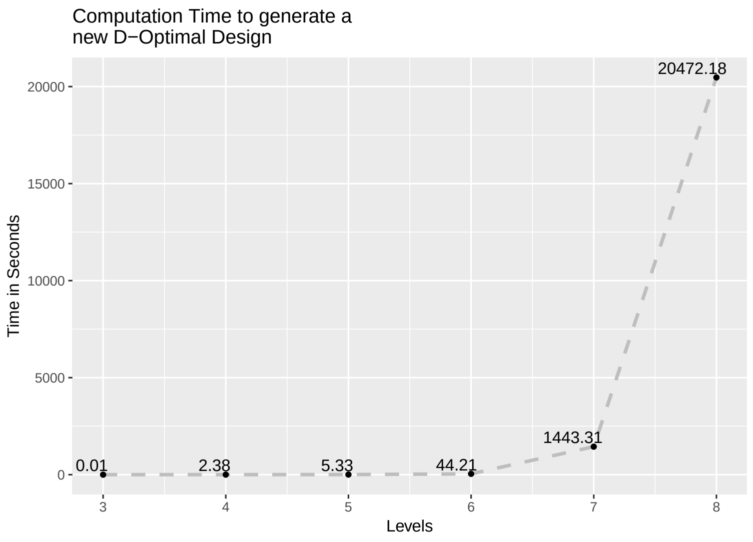

Figure 4: Computation time to generate a new D-Optimal design using Fedorov Algorithm. We keep all parameters constant, i.e. 7 attributes and 1000 alternatives, and only change the number of levels.

Notice that generating a D-Optimal design in Sawtooth Software with the same characteristics with 7 attributes and 3 number of levels each takes around 3 hours. Due to the high computational costs a comparison between the approach here presented and the one posed by SSI is unfeasible.

The Fedorov Algorithm is proved to be a fast algorithm for the problem in hand. But how efficient is it compared to a random design like those generated by SSI?

We can compare them by computing the D-Efficiency criteria, Equation 4.2, of each design and comparing one with the other, using the same parametrization as in Figure 4. As seen in Figure 5 the D-efficiency of the design generated by the Fedorov Algorithm is always larger to the D-efficiency of the random design. For instance, a 3^7 full factorial design, the Fedorov approach is a 0.37% more efficient than the random. It is clear that the larger the full factorial design, the larger discrepancy between methods, thus the better the Fedorov algorithm performs.

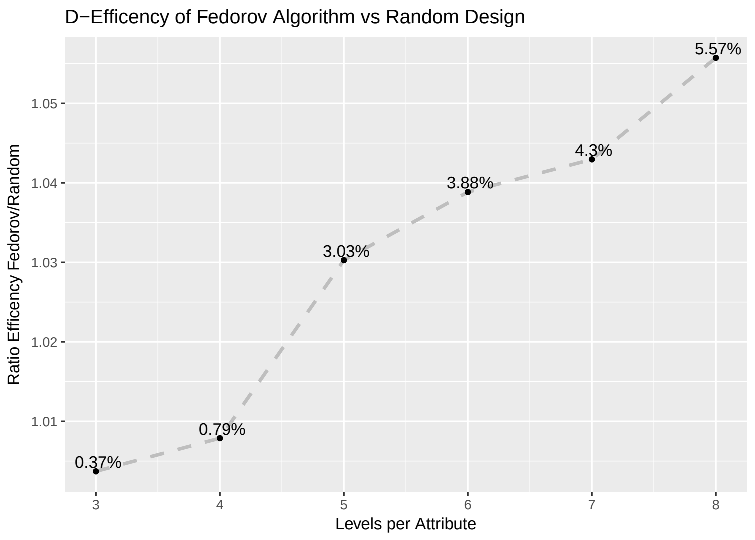

Figure 5: Ratio between D-Efficiency of $Fedorov/Random$. Fedorov Algorithm outperforms a random design like those generated by standard software packages.

In order to maximize the information from respondents we need a D-Optimal design. Instead of generating a new D-Optimal design on every iteration we just reuse the original one. Those rows that contained a removed level are exchanged using Fedorov Algorithm. This approach speeds up computations notably as we only need to evaluate few rows. In the simulation here presented we use 6 different tasks and 4 concepts per individual for a total of 75 respondents per iteration.

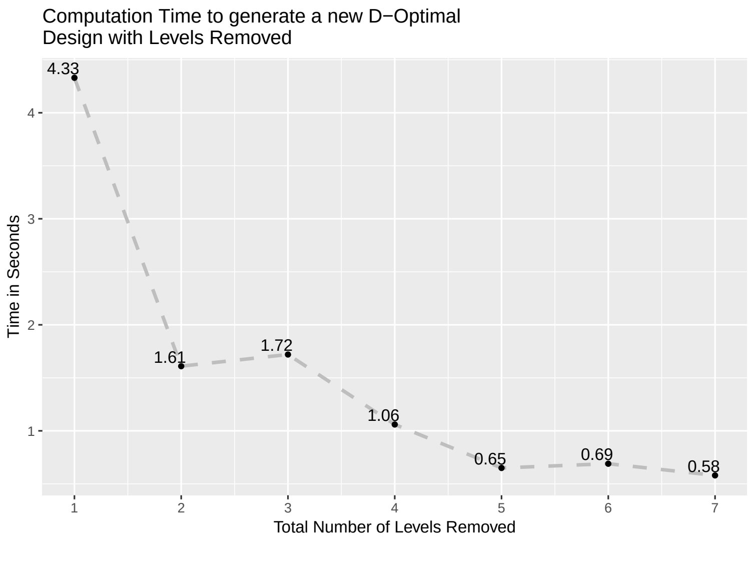

Figure 6: Computation time to generate D-Optimal designs when removing levels. The more levels removed the faster the algorithm as the design is smaller and fewer points need to be evaluated.

As one can see in Figure 6 the computation time to generate a new D-Optimal design departing from the original is very small. It takes less than a second to exchange the undesired alternatives with optimal ones. This is an important fact, because we manage to avoid the generation of completely new designs which tend to take longer to be created. By using the novel approach reported in this research we only need to evaluate few candidate points instead of evaluating all possible combinations.

6.2 Simulation

In order to test the algorithm we first need to set up few assumptions. We have to simulate the real individual utilities. As seen in Equation 4.4 we assumed a Normal Distribution with certain means and covariance matrix. The selection of such a distribution is not arbitrary. We use a Normal distribution because of its usefulness in a very wide range of situations; it is the most common distribution in nature; and has some properties which make it very special, for instance the Central Limit Theorem. In addition to the theoretical arguments, we can motivate the choice of individual utilities normally distributed, from empirical data. As explained in Section 5, we use 5 different datasets with real utilities from previous studies at SKIM. If we look at the distribution of individual utilities from these real data, we observe that the distributions are Normal with certain mean and covariance. In Figure 7 we can see the distribution of four different attributes with their respective levels plotted for four different datasets.
As we can see there is certain variation between distributions, e.g. ones are flatter, others more peaked, different mean etc. This comes to say that assuming Normal distributions for the simulated utilities is a natural and legitimate approach.

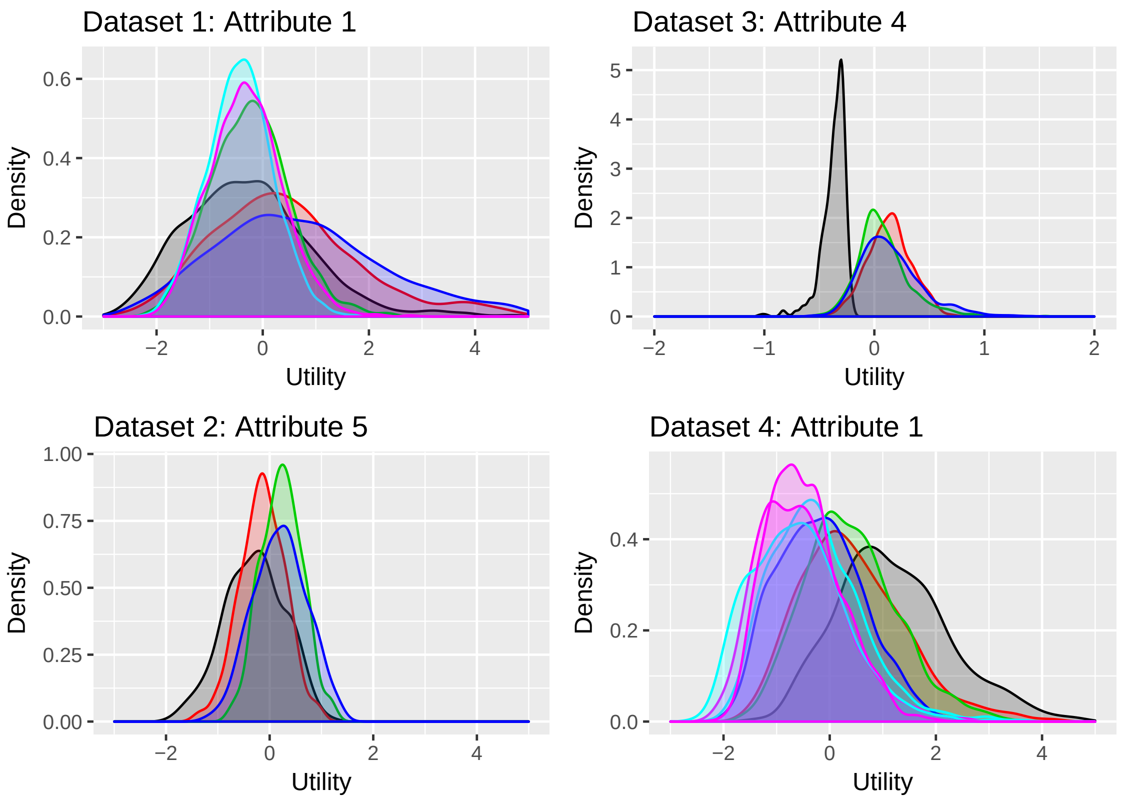

Figure 7: Distribution of individual utilities of specific attributes in four of the different real datasets. Distributions are approximately normal distributed.

First of all we simulate two slightly different settings. Firstly, we use a very sparse covariance matrix, with high variance then we compare with the same model but with smaller variance on the utilities. To assess the performance of our model we use the computation time and the rank of the final alternative for each individual with respect to the worst and best possible individual alternative. One can find the covariance matrix of both simulations in Appendix B, Equation B.1 and B.2.

In Figure 8 one can observe that the utility of the optimal product derived from the algorithm is close to each individuals' best alternative. In this case we considered 100 respondents per iteration, with a design that consists of 6 attributes with the following levels $v = \{6, 11, 4, 6, 4, 7\}$. We account for the high variance of the utilities by using the removal procedure presented in Equation 4.12. The whole algorithm needs about 50 seconds to find a candidate global optimum. Taking into account that there are 44,352 possibilities, we can say it is a relatively fast algorithm.

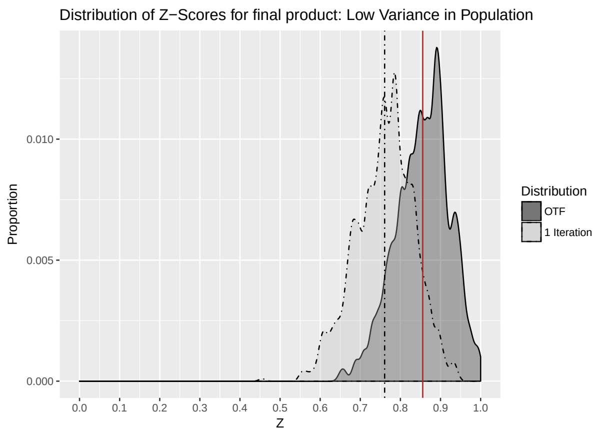

Figure 8: Distribution of the individual Z-Scores, where 0 is the worst possible product and 1 the best one. The red solid line is the mean for the OTF Algorithm and the dashed one the average Z-Score if we did not remove any level and just select the levels with highest utility. The final product has very high scores on the Z-Scale in this Low Variance case.

If we look at population level we see in Table 8 that the optimal alternative output by the Algorithm manages to converge to the best alternative except for Attribute 2 which predicts level number 9 when the true is the 6.

Table 8: Optimal product at population level compared to the best and worst possible alternatives for the Low Variance case.

	Best	Worst	Predicted	Correct
Attribute 1	5	3	5	✓
Attribute 2	6	10	9	
Attribute 3	2	4	2	✓
Attribute 4	3	2	3	✓
Attribute 5	3	4	3	✓
Attribute 6	1	3	1	✓

In case of high variance the Algorithm fails dramatically to converge to the population best alternative. This is caused by the fact that the variance is extremely high so the algorithm removes the correct levels when it should not do it. In Figure 9 one can see that the performance at individual level is very poor. In Table 9 we see that at population level we do not manage to converge to the optimal alternative as only two attributes are correctly identified as optimal.

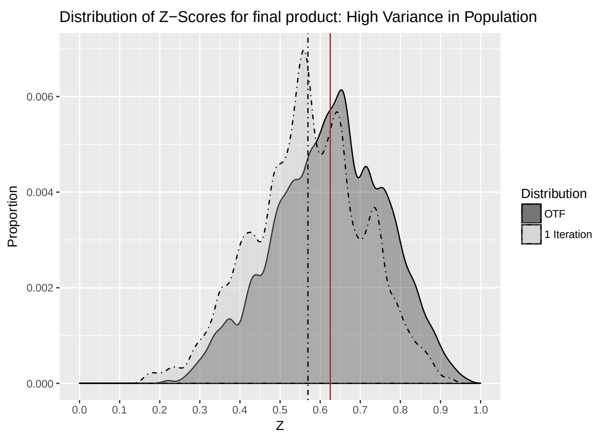

Figure 9: The scenario with a large covariance matrix yields worse results; Given that variance is larger, there is also more heterogeneity among respondents, thus it is more difficult to achieve a candidate global optimum. The red line is the average Z-Value for the OTF algorithm and the dashed one for the utility in case we did not remove any level.

Table 9: Optimal product at population level compared to the best and worst possible alternatives for the High Variance case.

	Best	Worst	Predicted	Correct
Attribute 1	4	6	4	✓
Attribute 2	1	3	8	
Attribute 3	3	2	4	
Attribute 4	1	2	5	
Attribute 5	4	3	4	✓
Attribute 6	2	4	6	

In order to further explore the capabilities of the algorithm here presented, we conduct a sensitivity analysis. To do so, we compare the average fit of the final product in the Z-Scale, that is how much the final product scores from 0 to 1, with respect to utilities with different variances. We start by creating 6 different attributes with the following levels $v = \{4, 5, 4, 6, 4, 3\}$. We generate the individual utilities by sampling random numbers from a normal distribution with mean μ, the interested reader can find them on Appendix B Equation B.4. For the variance we generate a diagonal covariance matrix, i.e. no cross-correlations between levels and attributes, but we modify the diagonal elements, that is the variance, increasing them iteratively. As seen in Figure 10, if we keep everything constant but we increase the variance of the real individual level utilities, the average fit decreases. So higher variance on the utilities leads to less optimal final products. That can be explained by the fact that heterogeneity is very large so it is more difficult to find a global optimum, in other words, a population best alternative.

If we look at the optimal product by population means, that is that product whose level utilities are highest per attribute, we see that it is never removed or it is removed in the very last iterations. This will depend on several factors such as the original D-Optimal design, the subsequent designs, and the overlap between levels.

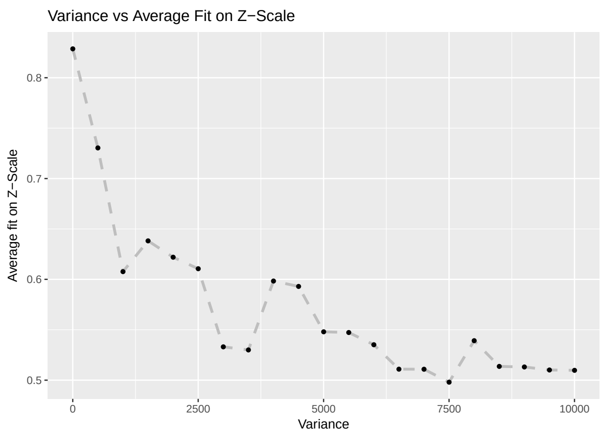

Figure 10: If we increase the variance of the simulated true utilities, the average fit of the final product decreases. The individual fit is very poor in very extreme cases.

A more general approach to understand if the results of the algorithm were produced by chance, we generate 6 attributes with the following levels $v = \{4, 5, 4, 6, 4, 3\}$ and assume a random uniform distribution for the mean of each level. In addition, we generate random utilities out of a multivariate normal distribution. By repeating this procedure 1,000 times we can compute a distribution of the average value of the Z-Scale on every iteration, thus how we expect to perform the algorithm in more general terms.

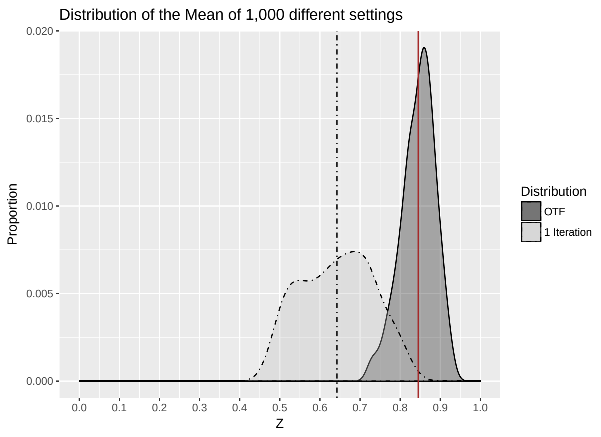

Figure 11: Distribution of the mean of the final product on the Z-Scale, using 1,000 random simulations of utilities. The algorithm here presented deviates a 15% on average from the most optimal product at individual level. If we only run one iteration the deviation is around 0.35%.

In Figure 11 we depict the distribution of means using 1,000 different utilities configurations. As you can see, we expect the mean value of the Z-Scale to be between 0.7 and 0.95 on average, being around 0.85 the most frequent value. That is only a 15% deviation of the final product with respect to the most preferred product possible. The implication is straightforward: if we use a standard CBC study and select those levels with highest utility we will obtain worse results than applying the CBC-OTF algorithm. Therefore, this Thesis poses a significant improvement in the field of CBC and Market Research when dealing with optimal products.

If we consider optimality at population level as in Table 10 we see the percentage of correct predicted levels is quite high: Attribute 3 is predicted correctly in 95.4% of the cases, Attribute 2 is predicted correctly in 66.7% of the time taking into account 1,000 iterations.

Table 10: Percentage of correct predicted levels with 1,000 different settings.

	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6
True Level	1	1	2	5	4	3
% Correct	89.3	66.7	95.4	72.1	71.2	79.5

Interestingly, the algorithm only converges to the optimal alternative in 35.8% of the cases. However, if we consider nearly-optimal allocations, those where less than two levels are incorrectly predicted, the percentage increases to 78.6% of correct.

This research uses individual and population level measures although some improvement in the population level measure could be done in order to increase accuracy. This paper intention is to set up the guidelines for further research.

6.3 Sensitivity Analysis

When working with the CBC-OTF Algorithm the researcher needs to take decisions regarding some of the settings. First of all, he needs to decide how many respondents he uses in every iteration. Another important element to analyze is if we could arrive to the optimal solution at individual level with fewer iterations. These two settings are paramount for practitioners and users of the algorithm.

To understand how these elements behave we design a sensitivity analysis i.e. holding everything constant, *Ceter Paribus*, just modify these settings to see how the changes occur. We use the usual approach of 6 tasks with 4 concepts each, and use the same utility means and covariance matrix as in B.1 and B.3. As can be seen in Figure 12, if we increase the number of respondents per iteration there is a slight improvement in the individual utilities measure or Z-Score. The marginal increase is very small over 50 respondents, it even gets staggered in terms of Z-Score.

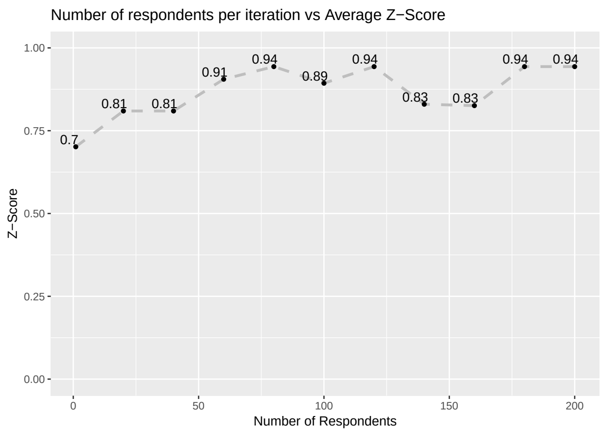

Figure 12: The marginal increase in the average Z-Score is decreasing when more respondents are used on each iteration.

Another setting a researcher can tweak is the total number of levels to be removed. It may happen that by removing a certain amount of levels we may also reach the optimal product. In Figure 13 one can see that the more levels we remove the higher the individual utility measured in terms of Z-Score.

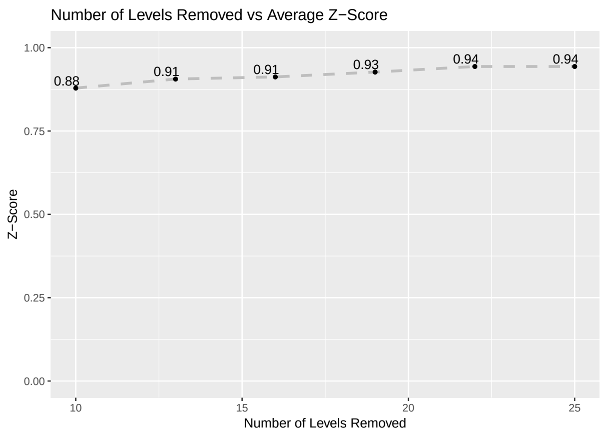

Figure 13: *Ceter Paribus*, the more levels removed the higher the Z-Score.

From this analysis we can propose some general guidelines for practitioners. Using around 50 respondents per iteration seems a reasonable amount of people to obtain precise enough results. In addition, if time is not an important constraint, we suggest to remove all the possible levels until only one alternative is left.

6.4 Empirical Application

As explained in the Data Section we use five different real datasets to evaluate how well our algorithm performs. The input individual utilities are assumed to be true ones. They come from a Hierarchical Bayes estimation for previous research projects for SKIM clients. Given that the present research is intended for future application, for each of the datasets analyzed we use the standard approach of SKIM for a CBC study of these characteristics: 6 different tasks with 4 different concepts each one.

6.4.1 Dataset 1

Recall the main setting of this scenario:

- Attributes: 8
- Levels: $v_1 = \{6, 3, 7, 5, 2, 5, 5, 6\}$
- Individuals: 2057

For this scenario, we use 75 individuals per iteration when fitting the logit model. We take into account the variance of the parameter estimates to assess which levels are not informative enough. The computing time needed for the algorithm to find a candidate global optimum with the current settings

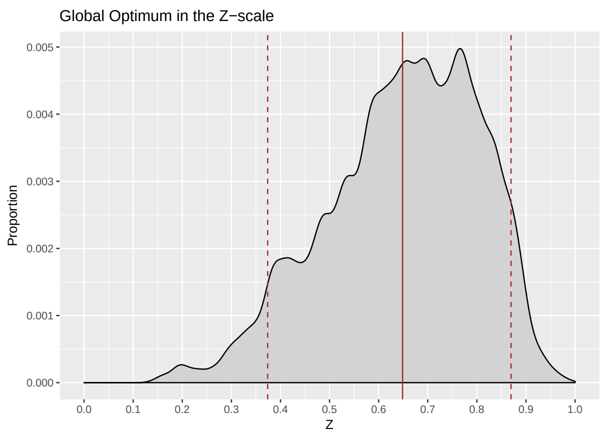

Figure 14: Distribution of the individual ranked utilities, where 0 is the worst possible product and 1 the best one. The red solid line is the mean and the dashed ones the 5% and 95% quantiles. The final product has very high utility, over 0.75, for more than 40% of the sample.

We observe in this case that the average location of the global product in the Z scale is 0.64 and the 5% and 95% quantiles are between 0.37 and 0.87. This means that we manage to obtain quite good results for the general population. We have to consider that individual preferences are not uniform across population, so finding a product that maximizes utility for every individual, i.e. $z_i = 1, \forall i = 1, \ldots, n$, is something unfeasible. In addition, the computation time of the whole algorithm is 103.8 seconds and 19 iterations. The interested reader can find in appendix B Table 16 the progression of level removal iteration by iteration.

As explained before, this approach only considers main effects. However, in this dataset we know the presence of price in one of the variables which probably may lead to sub-optimal allocations. For instance, we may end up selecting those attributes that only consider free features, because they yield the maximum utility compared to those that cost money. As we do not have further information about the data we cannot confirm this hypothesis.

Regarding at population level we see in Table 11 that the algorithm here presented fails to find the most optimal product at average population utility. Only two attributes of the final alternative coincide with the levels with highest utility. There might be different reasons that make the Algorithm fail. Most likely it is due to the fact of cross effects between levels and probably the presence of a variable such as price.

Table 11:

	Best	Worst	Predicted	Correct
Attribute 1	3	6	2	
Attribute 2	1	2	1	✓
Attribute 3	4	1	5	
Attribute 4	1	3	2	
Attribute 5	1	2	1	✓
Attribute 6	3	4	2	
Attribute 7	3	5	4	
Attribute 8	6	3	4	

6.4.2 Dataset 2

Main setting of this scenario:

- Attributes: 5
- Levels: $v_2 = 7, 3, 3, 4, 4$
- Individuals: 242

For this scenario, we use 50 individuals per iteration when fitting the logit model. We want to obtain reliable estimates quickly. The total computation time using such a setting is only 16 seconds and 8 iterations to obtain a candidate global optimum. The factors that contribute to such a high speed of convergence are that the full factorial design is not large, only 1008 possible alternatives in total, and we use only 50 individuals per iteration.

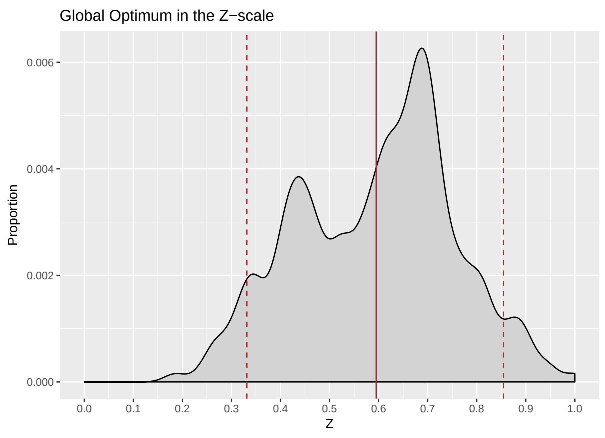

Figure 15: Distribution of the individual ranked utilities, where 0 is the worst possible product and 1 the best one. The red solid line is the mean and the dashed ones the 5% and 95% quantiles. We find a very large peak at 0.72 in the Z-Scale.

In Figure 15 we can find the distribution of the Z value. Both mean and median are 0.6 and the 5% and 95% quantile are between 0.33 and 0.85. The distribution is clearly bimodal, with a peak at around 0.72 and a smaller peak at 0.43. Although there is room for improvement, results invite to optimism. We manage to get very good utility levels for a large amount of individuals, and very few people have very low values in their utilities. As we can see, almost there are not individuals below 0.2.

If we consider optimality at population level as in Table 12 not all the levels converge to the optimal point. Attribute 1 and Attribute 5 converge to a suboptimal allocation.

Table 12: Three out five levels converge to the optimal level. The product is nearly optimal.

	Best	Worst	Predicted	Correct
Attribute 1	3	6	7	
Attribute 2	2	3	2	✓
Attribute 3	3	1	3	✓
Attribute 4	2	1	2	✓
Attribute 5	4	2	3	

6.4.3 Dataset 3

Main setting of this scenario:

- Attributes: 9
- Levels: $v_3 = \{3, 3, 5, 4, 3, 4, 3, 4, 5\}$
- Individuals: 817

This dataset, which we have no information at all of what it is about, has very balanced attributes, in the sense that they only have 3 to 5 levels. We use 75 respondents per iteration to obtain parameter estimates. Taking into account that around 1200 respondents is a standard figure for an average CBC study at SKIM; 75 respondents per iteration considering that we will probably have around 10-15 iterations is a reasonable size.

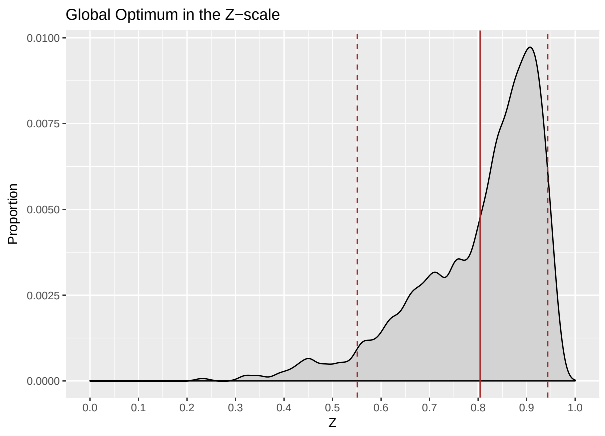

Figure 16: Distribution of the individual ranked utilities, where 0 is the worst possible product and 1 the best one. The red solid line is the mean and the dashed ones the 5% and 95% quantiles. The great majority of the individuals almost maximize their utility with the final product.

The whole process takes 55.5 seconds and 14 iterations to find a candidate global optimum. As one can see in Figure 16 the utility of the candidate global optimum is excellent for a large proportion of individuals. A mean of 0.8 and median 0.84 make this algorithm an appropriate approach to arrive to the best product. Over 50% of the individuals only deviate ~ 15% from their most possible optimal product.

However, if we consider the population level measure we find that there is not convergence at all as in Table 13 is shown. Not even a single level of the optimal alternative output by the Algorithm manages to be the one with highest true utilities.

Table 13: The Algorithm does not manage to converge to the optimal alternative.

	Best	Worst	Predicted	Correct
Attribute 1	3	1	1	
Attribute 2	1	3	3	
Attribute 3	1	4	4	
Attribute 4	2	3	4	
Attribute 5	1	3	2	
Attribute 6	3	1	1	
Attribute 7	2	3	1	
Attribute 8	1	2	2	
Attribute 9	2	4	1	

6.4.4 Dataset 4

Main setting of this scenario:

- Attributes: 6
- Levels: $v_4 = \{8, 17, 4, 5, 4, 3\}$
- Individuals: 1723

This is an interesting case because there is an attribute with many levels, 17. This may have a relatively high attribute importance because of the fact that utilities of each level will be probably close to each other. The process takes in total 53.5 seconds and 19 iterations using 75 respondents per iteration. Again, the reader can find the progression of the removal of levels on Appendix B Table 19.

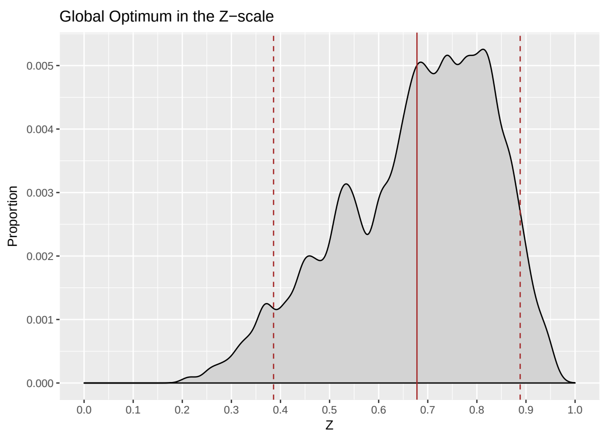

Figure 17: Distribution of the individual ranked utilities, where 0 is the worst possible product and 1 the best one. The red solid line is the mean and the dashed ones the 5% and 95% quantiles. Over 50% of the sample deviate less than 30% from the most optimal possible combination.

The mean of the Z-Scale is at 0.68 and the median at 0.7. This means that results are good for a large proportion of the respondents, over 50% of the individuals are only a 30% off from the most optimal product.

In terms of optimality at population level we find that the Algorithm converges in half of the levels as shown in Table 14.

Table 14: At population level the Algorithm predicts correctly half of the levels.

	Best	Worst	Predicted	Correct
Attribute 1	3	6	4	
Attribute 2	5	2	8	
Attribute 3	1	2	1	✓
Attribute 4	2	4	2	✓
Attribute 5	1	2	3	
Attribute 6	1	3	1	✓

6.4.5 Dataset 5

Main setting of this scenario:

- Attributes: 8
- Levels: $v_5 = \{12, 3, 2, 3, 2, 3, 5, 6\}$
- Individuals: 273

In this case we have again an attribute with too many levels compared to the other attributes. We decide to ask 250 respondents per iteration. Notice that the original utilities are only 273 so the same respondents utilities may be reused in different iterations. This configuration is to proof how fast the algorithm can be and deal with very large designs in a relatively short time. The total computation time to find a candidate global optimal product is 110.1 seconds and 15 iterations. This is a very fast algorithm as in the full factorial design we can find 38880 different alternatives, within one minute and a half we can empirically select the best alternative.

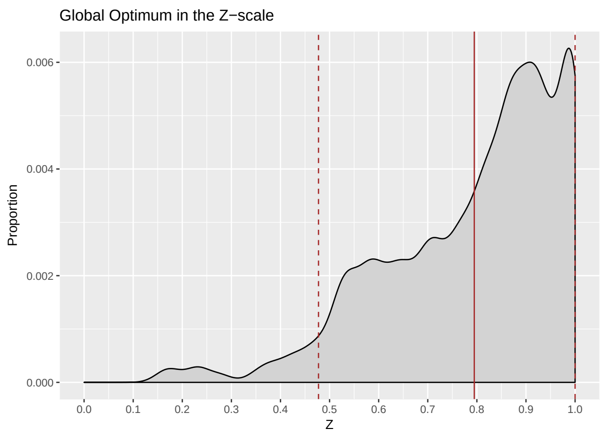

Figure 18: Distribution of the individual ranked utilities, where 0 is the worst possible product and 1 the best one. The red solid line is the mean and the dashed ones the 5% and 95% quantiles. Results are excellent, individual utilities for the final product rank very high.

As we can see in Figure 18 results are astonishing. For 35% of the respondents we manage to create a product that deviates less than 10% from their most optimal alternative. In fact, the 95% quantile is at 1, and the 5% at 0.47.

However, if we consider the population level measure it predicts correctly half of the levels as one can see in Table 15.

Table 15: At population level half of the levels are correctly identified.

	Best	Worst	Predicted	Correct
Attribute 1	3	10	11	
Attribute 2	1	2	3	
Attribute 3	1	2	1	✓
Attribute 4	3	2	3	✓
Attribute 5	1	2	1	✓
Attribute 6	2	3	3	
Attribute 7	1	2	5	
Attribute 8	1	3	1	✓

7 Software Implentation

The algorithm presented in this research only makes sense if it is applicable. For that purpose, an R package has been developed. The library is called *OTF* and it contains several functions to run the analysis with simulated data. It can contain completely simulated utilities from user-defined distributions, or the user can also source a file with utilities from previous studies. The package includes instructions on how to use the functions.

The functions included are the following:

- fedorovGenerator: Allows to create D-Optimal designs using Fedorov exchange algorithm. The user has to specify different parameters such as the number of attributes and levels.
- startingDesign: This function generates subsequent D-Optimal designs with the specified levels removed departing from an initial D-Optimal design.
- CBC_Generator: Generates a CBC study assigning different tasks and concepts to every individual.
- DependentVariable: Generates a dependent variable from the simulated individual utilities.
- CBC_Utils: We obtain a full CBC study with every concept shown on every task for each individual and the concept she selected.
- ModelEstimation: Fitting of a multinomial logit model.
- RemoveLevels: The function makes the level assessment and outputs those levels that have to be removed on every iteration.
- MainSimulator: This function is the core of the package. It combines all previous functions to make a very simple simulator which requires very few inputs. It starts from a full factorial design and it ends up giving as output the most optimal product and how the iterative process was carried out.
- LevelAssessment: We use this function to evaluate how optimal the final product is compared to other alternatives. It uses the Z-Scale as method of evaluation and it outputs summary statistics of the distribution.

The package has been coded in such a way that someone with little experience in R and Conjoint Analysis can use it easily. All the functions have been written in a way that are easily adaptable for further implementations and development.

8 Conclusions

Deciding which product should be commercialized and in which shape is not a trivial question. This research has focused on developing a new modeling framework which is capable of providing the best alternative, the most preferred product by the great majority of consumers depending on individual preferences. An implementation of Choice Based Conjoint Analysis has been exposed. This approach tries to iteratively remove the undesired levels so as to reduce the space of alternatives. The goal is to to learn about respondents to optimize the following tasks they have to complete. This leads to higher efficiency and increased accuracy in the parameter estimates.

The algorithm starts with the generation of a design. As opposed to common statistical packages that generate designs randomly, we use the D-Efficiency criteria by recurring to the Fedorov Algorithm to generate an optimal design. Although the computation time to generate such a design increases exponentially with the size of the full factorial design, it is still very fast, below 45 seconds for a 6^7 design for instance, and it has an increased efficiency compared to randomized designs. The use of a very optimal initial design and the subsequent modification of this for later iterations leads to an extremely efficient design generation in terms of D-Efficiency and computation time.

Given that one of the main concerns of the framework here presented was the speed of convergence towards a candidate global optimum, we rejected the use of Hierarchical Bayes model in favor of a multinomial logit model.

An important feature of this novel algorithm is the use of the integrated part between the so-called Pseudo-density functions. This allows us to select the worst two levels per attribute in terms of utility. They are selected because we want to remove on every iteration those levels that we know for sure that people dislike. Thus comparing between worst and best levels does not provide additional information in the removal process. The fact of using a probability function to decide which exactly level should be removed from a specific attribute helps to generalize the model.

The generation of the subsequent designs, as proved by simulations and empirical application, is extremely fast, less than 3 seconds. This feature makes the algorithm very special: we can converge to a candidate global optimum very quickly.

In order to evaluate how optimal the final product is we create a Z-Scale. This scale ranks from 0 to 1, where 0 is the worst possible individual product and 1 the best. The final product given by the OTF algorithm falls inside this range. In addition, a population best measure is included in order to assess the optimality at population level.

Finally an empirical part with simulated and real data was carried out. We proved that higher variance in the individual utilities leads to less optimal final products. This is greatly caused by high heterogeneity across individuals. By generating 1,000 random distributions with different configuration on the mean and covariance matrix, we are able to evaluate how the algorithm performs. We found out that the final alternative deviates no more than 15% from the most optimal product at individual level. A great amount of individuals deviate less than 5%. This comes to say that on average the algorithm is able to find an extremely optimal product for a great majority of the population. Notice that achieving a 100% accuracy deems as impossible, otherwise it would mean that all the respondents are almost the same and there is no heterogeneity across preferences on individuals.

Results regarding the empirical application with SKIM's anonymous data are very positive. We find that deviations from the most optimal product at individual level are relatively small, thus we manage to find a very good product for a great majority of the respondents.

This approach poses a new line of research in the Conjoint Analysis field. It sets the foundations for further exploring the capabilities of such algorithmic approaches to the optimization of products. Although results are very optimistic and promising there is still room for improvement and further research.

9 Limitations

This research represents a first step on this line of research. However, we are aware of the practical limitations that this approach has.

If the full factorial design is too large, millions of unique alternatives, we may face computational issues to generate new designs as this process may take up to several hours. Another important issue that needs to be addressed is the fact that only main effects can be computed. This implies that attributes that may affect severely the estimates due to the correlation with other attributes may not be included. A clear example would be a variable such as *price*: an attribute which theoretically should have larger utility, like better Internet connection, is worse because the price associated is too high.

Given that this is the first paper considering such an approach and sets the foundations for further developments, the algorithm does not allow to have real respondents on real time. Thus all results that can be extracted at this stage come from simulation schemes either with simulated or empirical utilities.

10 Further Research

We suggest that further improvement and research can be done in improving the generation of faster and more efficient designs. Although this is a very difficult task because the intrinsic efficient nature of the Fedorov Algorithm, some other algorithms may lead to different results.

An interesting line of research that was considered for this paper but was finally dismissed consists of removing one level per attribute on every iteration instead of one or two levels in total per iteration. Undoubtedly this would decrease the number of iterations hence increase the speed of the whole algorithm, but it may eventually lead to worse results. The present research intends to set the foundations on the field of CBC-OTF thus we leave for further research such a line of investigation.

We recommend to inspect the possibilities of writing the code in a compiled language due to is increased speed. The fact that the programming language used has been R probably decreases the computation speed.

References

Atkinson, A. C. and Donev, A. N. (1989). The construction of exact d-optimum experimental designs with application to blocking response surface designs. *Biometrika*, pages 515–526.

Betancourt, M. (2010). A bayesian approach to histogram comparison. *arXiv preprint arXiv:1009.5604*.

Borgwardt, K. M. and Ghahramani, Z. (2009). Bayesian two-sample tests. *arXiv preprint arXiv:0906.4032*.

Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

Chasalow, S. D. (1992). *Exact optimal response surface designs with random block effects*. University of California, Berkeley.

Chrzan, K. and Orme, B. (2000). An overview and comparison of design strategies for choice-based conjoint analysis. *Sawtooth software research paper series*.

Cook, R. D. and Nachtsheim, C. J. (1980). A comparison of algorithms for constructing exact d-optimal designs. *Technometrics*, 22(3):315–324.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242.

Donev, A. and Atkinson, A. (1988). An adjustment algorithm for the construction of exact d-optimum experimental designs. *Technometrics*, 30(4):429–433.

Fedorov, V. V. (1972). *Theory of optimal experiments*. Elsevier.

Galil, Z. and Kiefer, J. (1980). Time-and space-saving computer methods, related to mitchell's detmax, for finding d-optimum designs. *Technometrics*, 22(3):301–313.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2014). *Bayesian data analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL, USA.

Green, P. E., Krieger, A. M., and Agarwal, M. K. (1991). Adaptive conjoint analysis: Some caveats and suggestions. *Journal of Marketing Research*, pages 215–222.

Green, P. E. and Rao, V. R. (1971). Conjoint measurement for quantifying judgmental data. *Journal of Marketing research*, pages 355–363.

Green, P. E. and Srinivasan, V. (1978). Conjoint analysis in consumer research: issues and outlook. *Journal of consumer research*, 5(2):103–123.

Green, P. E. and Srinivasan, V. (1990). Conjoint analysis in marketing: new developments with implications for research and practice. *The Journal of Marketing*, pages 3–19.

Green, P. E. and Wind, Y. (1975). New way to measure consumers' judgments. *Harvard Business Review 53*, pages 107–117.

Halme, M. and Kallio, M. (2011). Estimation methods for choice-based conjoint analysis of consumer preferences. *European Journal of Operational Research*, 214(1):160–167.

Huber, J., Bacon, L., and Worldwide, N. (2003). Adaptive choice based conjoint analysis.

Huber, J. and Train, K. (2001). On the similarity of classical and bayesian estimates of individual mean partworths. *Marketing Letters*, 12(3):259–269.

Huber, J. and Zwerina, K. (1996). The importance of utility balance in efficient choice designs. *Journal of Marketing research*, pages 307–317.

Johnson, F. R., Lancsar, E., Marshall, D., Kilambi, V., Mühlbacher, A., Regier, D. A., Bresnahan, B. W., Kanninen, B., and Bridges, J. F. (2013). Constructing experimental designs for discrete-choice experiments: report of the ispor conjoint analysis experimental design good research practices task force. *Value in Health*, 16(1):3–13.

Johnson, M. E. and Nachtsheim, C. J. (1983). Some guidelines for constructing exact d-optimal designs on convex design spaces. *Technometrics*, 25(3):271–277.

Lilliefors, H. W. (1967). On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, 62(318):399–402.

Louviere, J. J. (1988). *Analyzing decision making: Metric conjoint analysis.* Sage.

Lucas, J. M. (1974). Optimum composite designs. *Technometrics*, 16(4):561–567.

Massey Jr, F. J. (1951). The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78.

Meyer, R. K. and Nachtsheim, C. J. (1995). The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*, 37(1):60–69.

Mitchell, T. J. (1974). An algorithm for the construction of "d-optimal" experimental designs. *Technometrics*, 16(2):203–210.

Näther, W. (1985). Exact designs for regression models with correlated errors. *Statistics*, 16(4):479–484.

Nguyen, N.-K. and Miller, A. J. (1992). A review of some exchange algorithms for constructing discrete d-optimal designs. *Computational Statistics & Data Analysis*, 14(4):489–498.

Orme, B. (2002). Formulating attributes and levels in conjoint analysis. *Sawtooth Software Research Paper*, pages 1–4.

Rao, V. R. et al. (2014). *Applied conjoint analysis.* Springer.

Reliasoft Corporation (2015). *Experiment Design & Analysis Reference.*

Sandor, Z. and Wedel, M. (2005). Heterogeneous conjoint choice designs. *Journal of Marketing Research*, 42(2):210–218.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Toubia, O., Hauser, J. R., and Simester, D. I. (2004). Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research*, 41(1):116–131.

Walker, S. H. and Duncan, D. B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179.

Wheeler, R. E. (2004). Comments on algorithmic design. *Vignette accompanying package AlgDesign.*

Wilcox, R. R., Erceg-Hurn, D. M., Clark, F., and Carlson, M. (2014). Comparing two independent groups via the lower and upper quantiles. *Journal of Statistical Computation and Simulation*, 84(7):1543–1551.

Wittink, D. R., Krishnamurthi, L., and Reibstein, D. J. (1990). The effect of differences in the number of attribute levels on conjoint results. *Marketing Letters*, 1(2):113–123.

Wynn, H. P. (1972). Results in the theory and construction of d-optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 133–147.

Zwerina, K., Huber, J., and Kuhfeld, W. F. (1996). A general method for constructing efficient choice designs. *Durham, NC: Fuqua School of Business, Duke University.*

A Appendix

MLE Multinomial Logit Model

$$\log\left(\frac{\pi_{ij}}{\pi_{iJ}}\right) = \log\left(\frac{\pi_{ij}}{1-\sum_{j=1}^{J-1}\pi_{ij}}\right) = \sum_{k=0}^{K} x_{ik}\beta_{kj} \tag{A.1}$$

for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, J-1$

Solving for π_{ij} we have

$$\begin{aligned} \pi_{ij} &= \frac{exp(\sum_{k=0}^{K}\beta_{kj}x_{ik})}{1+\sum_{j=1}^{J-1} exp(\sum_{k=0}^{K}\beta_{kj}x_{ik})} \qquad \text{j < J} \\ \pi_{iJ} &= \frac{1}{1+\sum_{j=1}^{J-1} exp(\sum_{k=0}^{K}\beta_{kj}x_{ik})} \end{aligned} \tag{A.2}$$

The Likelihood function is then given by

$$\prod_{i=1}^{N}\prod_{j=1}^{J-1} exp\left(y_{ij}\sum_{k=0}^{K}\beta_{kj}x_{ik}\right)\left(1+\sum_{j=1}^{J-1} exp(\sum_{k=0}^{K}\beta_{kj}x_{ik})\right)^{-n_i} \tag{A.3}$$

If we take logs we obtain the log likelihood function which is defined as

$$l(\beta) = \sum_{i=1}^{N}\sum_{j=1}^{J-1}\left(y_{ij}\sum_{k=0}^{K}\beta_{kj}x_{ik}\right) - n_i\log\left(1+\sum_{j=1}^{J-1} exp\left(\sum_{k=0}^{K}\beta_{kj}x_{ik}\right)\right) \tag{A.4}$$

By maximizing this equation it follows that

$$\frac{\partial l(\beta)}{\partial\beta_{kj}} = \sum_{i=1}^{N} y_{ij}x_{ik} - n_i\pi_{ij}x_{ik} \tag{A.5}$$

Thus we have $(J-1)(K+1)$ equations and we need to solve for each β_{kj}.

B Appendix

$$\begin{aligned} \Sigma_{Low} = diag\{&5,6,5,9,2,9,6,9,3,3,2,2,4,2,9,7,6,6,9,\\ &2,8,3,9,3,9,10,2,3,6,1,5,6,1,1,2,1,9,3\} \end{aligned} \tag{B.1}$$

$$\begin{aligned} \Sigma_{High} = diag\{&1797,3399,3502,3690,1887,1356,5372,3294,1223,4039,4493,4390,\\ &3362,4976,2569,3872,4320,2295,1867,5404,3427,5153,2113,2684,3679,\\ &4381,4250,3115,1792,1501,3776,4002,5496,4107,1684,4952,5383,2682\} \end{aligned} \tag{B.2}$$

$$\begin{aligned} \mu_{Low} = \{&15,-15,14,7,-13,-7,161,69,143,87,163,162,132,110,74,150,65,\\ &3,15,11,17,6,-18,-8,0,-11,-13,6,-15,13,-8,7,-12,6,17,1,-2,2\} \end{aligned} \tag{B.3}$$

$$\begin{aligned} \mu_{High} = \{&15,1,-1,-8,142,143,55,102,107,-8,6,-18,-17,\\ &-2,0,-4,-4,8,16,12,5,-4,16,9,-10,15\} \end{aligned} \tag{B.4}$$

Table 16: Progression of level removal iteration by iteration for Dataset 1.

	X1	X2	X3	X4	X5	X6	X7	X8
1	6	3	7	5	2	5	5	6
2	6	3	6	5	2	5	5	6
3	6	3	6	5	2	5	4	6
4	6	3	6	5	2	5	4	5
5	5	2	6	5	2	5	4	5
6	5	2	4	5	2	5	4	5
7	5	2	4	4	2	5	4	5
8	5	2	4	4	2	5	3	5
9	4	2	4	4	2	5	3	5
10	4	2	4	4	2	3	3	5
11	4	2	3	4	2	3	3	4
12	3	2	3	4	2	3	3	4
13	3	2	3	3	2	3	3	3
14	2	2	3	3	2	3	3	3
15	2	2	2	3	2	3	3	3
16	2	2	2	3	2	3	3	2
17	2	2	2	3	2	3	2	2
18	2	2	2	2	2	3	2	2
19	2	2	2	2	2	2	2	2

Table 17: Progression of level removal iteration by iteration for Dataset 2.

	X1	X2	X3	X4	X5
1	7	3	3	4	4
2	6	3	3	4	3
3	5	3	3	3	3
4	4	3	2	3	3
5	3	3	2	3	2
6	2	3	2	3	2
7	2	2	2	3	2
8	2	2	2	2	2

Table 18: Progression of level removal iteration by iteration for Dataset 3.

	X1	X2	X3	X4	X5	X6	X7	X8	X9
1	3	3	5	4	3	4	3	4	5
2	3	2	4	4	3	4	3	4	5
3	3	2	4	4	3	4	2	4	4
4	3	2	4	4	2	4	2	4	4
5	3	2	4	4	2	4	2	4	3
6	3	2	4	4	2	3	2	4	3
7	3	2	3	4	2	3	2	4	3
8	3	2	3	4	2	3	2	3	2
9	3	2	3	3	2	3	2	3	2
10	3	2	3	2	2	3	2	3	2
11	3	2	2	2	2	3	2	3	2
12	3	2	2	2	2	2	2	3	2
13	3	2	2	2	2	2	2	2	2
13	2	2	2	2	2	2	2	2	2

Table 19: Progression of level removal iteration by iteration for Dataset 4.

	X1	X2	X3	X4	X5	X6
1	8	17	4	5	4	3
2	8	16	4	5	3	3
3	8	15	4	5	3	2
4	7	14	4	5	3	2
5	7	13	4	4	3	2
6	6	12	4	4	3	2
7	6	11	4	4	2	2
8	6	10	3	4	2	2
9	6	9	2	4	2	2
10	5	8	2	4	2	2
11	4	7	2	4	2	2
12	4	6	2	4	2	2
13	4	4	2	4	2	2
14	3	4	2	4	2	2
15	2	4	2	4	2	2
16	2	3	2	4	2	2
17	2	3	2	3	2	2
18	2	3	2	2	2	2
19	2	2	2	2	2	2

Table 20: Progression of level removal iteration by iteration for Dataset 5.

	X1	X2	X3	X4	X5	X6	X7	X8
1	12	3	2	3	2	3	5	6
2	11	3	2	2	2	3	5	6
3	10	3	2	2	2	2	5	6
4	9	3	2	2	2	2	5	5
5	8	3	2	2	2	2	5	4
6	7	2	2	2	2	2	5	4
7	6	2	2	2	2	2	5	4
8	6	2	2	2	2	2	4	4
9	5	2	2	2	2	2	3	4
10	5	2	2	2	2	2	2	4
11	4	2	2	2	2	2	2	4
12	4	2	2	2	2	2	2	3
13	4	2	2	2	2	2	2	2
14	3	2	2	2	2	2	2	2
15	2	2	2	2	2	2	2	2