Product Design Optimization Via Conjoint Analysis

Derek Zoutendijk

March 8, 2019

Derek Zoutendijk
411370

Abstract

In this research, the product design optimization is addressed. The objective is to maximize an objective value such as margin, revenue or market share by composing a product based on multiple attributes. The Nested Partition (NP) method in combination with Genetic Algorithms (GA) and a machine learning method (CTREE) is used to optimize margin for a hairdryer manufacturer. Methods lead to high increases in margin compared to the current scenario in the hairdryer market.

Contents

1 Introduction

In product design optimization, products are composed such that objective values as revenue or market share are maximized. As new products are introduced very often, it is important for companies maintain their competitive position and optimize their own products. Composing an optimal product can be very complex. A simple product already possesses many different aspects or attributes such as price, colour, shape and size. For these attributes, we can choose from many different options or levels. To optimize market share, we want to compose the product in such a way that a customer chooses ours over competitor products. Blauw Research estimated the added utility to a product of an attribute level per respondent (customer) using a conjoint analysis. It is then possible to calculate the total utility of a product given any selection of attribute levels. If this is also done for the products of competitors, we can then make predictions on the probability of customers choices and eventually the profit of company. It is therefore the goal to select attribute levels in such a way that a given objective value is maximized.

As Blauw Research has multiple clients, it is important that the optimization model is applicable to a wide variety of problem instances. For some companies costs are included and the margin on a product has to be optimized, for others we are concerned with revenue or even market shares. For these different objectives, it should be possible to include multiple products in the optimization. Companies sometimes want to introduce new products into existing markets where they already offer products. It should thus be possible to optimize the profit of the entire product line. Due to development costs and avoiding risks, companies do not want to change many products in a product line. Therefore, we focus on adjusting a single product. Also, the model must be able to handle constraints and the possibilities of continuous and discrete variables have to be incorporated. Thus as stated before, a general optimization method is required.

In this research, we propose the use of a Nested Partition (NP) method to solve the product design problem. To enhance performance, the NP method is extended with Genetic Algorithms (GA) and a version of a Greedy Search (GS). The version of GS is based on machine learning methodology.

We now present an outline for this paper. In Section 2 we review previous work on the topic of product design optimization. Then, in Section 3 we give a more detailed description of the problem and in Section 4 we present a concise data analysis. Next, our optimization method is explained in Section 5. Computational experiments and results are shown in Section 6 and in Section 7 we draw conclusions and discuss possible further research.

2 Literature review

Our research combines different methods to solve the product design problem. For that reason, we will briefly discuss some academic references related to our work. We conclude by presenting a small survey of other research on product design optimization.

Conjoint Analysis
For estimating utilities of products, Blauw Research performs a conjoint analysis. Respondents represent the customers and are asked to choose between a few variants of a product. Repeating this gives an indication on which attributes and attribute levels affect the decision of buying a product. The term conjoint analysis was first introduced by Luce and Tukey [1964], but applications on product design optimization gained its popularity after Green et al. [1981]. Here, optimization is done by an enumerating procedure as the size of the problem instance was not an instance. Green et al. [2001] previse more complex applications for conjoint analysis, such as optimization of profits.

Complexity
In multiple instances we encountered, the number of possible designs are more than a few billion, thus enumerating over all these solutions is not an option. The product design problem can be formulated as a mixed integer program. However, the problem is shown to be NP-hard by Kohli and Krishnamurti [1989]. Therefore, we have to resort to heuristics.

Dynamic Programming
Kohli and Krishnamurti [1989] propose a Dynamic Programming (DP) approach. Product attributes represent stages with attribute levels as states. A product is iteratively formed. In a certain stage j, one calculates the best partial product profiles for each level of attribute $j+1$. These profiles are then again used for forming new product profiles in the next stage, thus for the next attribute. Following this results in a complete product.

Nested Partition
A similar method to DP in solving the product design problem is the NP method proposed by Shi et al. [2001]. Here again, attribute levels are selected iteratively until a complete product profile is formed. A level is selected based on how promising it is that the optimal solution contains that level. Contrary to DP, the NP method is shown to be globally convergent in finite time (Shi et al. [1999]). In improving the estimation of the promisingness of an attribute level, Shi and Ólafsson [2000] combine the NP method with GA. Stretching this idea even further, Shi et al. [2001] also include a GS heuristic.

Genetic Algorithms
A more popular method for the product design problem are GA (Jiao et al. [2007]). GA are based on natural selection. Given an initial solution set, we take a subset of solutions with highest objective values. Within this new set, couples of solutions are

formed that together will generate offspring. This is usually done by crossover and mutation. In the product design problem, crossover can be performed by swapping levels for some attributes within a couple. Mutation occurs after crossover and consists of randomly changing an attribute. The initial solution can be formed using any method and is therefore suited to be used in combination with other techniques (Balakrishnan and Jacob [1996]).

Greedy Search
Next to DP, Kohli and Krishnamurti [1989] also propose a shortest-path heuristic for a graph representation of the problem. This boils down to a GS, where for each attribute the level with highest aggregate utility over all respondents is selected. Solutions are shown to be close to optimal for simulated instances. However, instances are small and the GS could perform worse on larger cases. Moreover, for optimizing revenue or profit, this greedy method can result in bad solutions when selecting levels with high utilities leads to low prices or high costs.

Survey
We now discuss other research on product design optimization and explain why it is less applicable to our problem. Belloni et al. [2008] present a comparison between methods of optimizing an entire product line. The method closest to NP presented in this paper is a DP algorithm, which is shown to be outperformed by GA, Simulated Annealing and a Product-Swapping heuristic. In this paper, the problem consists of six attributes with only two levels and a price attribute with seven levels. Our problem instance is larger and we focus on the optimization of a single product instead of the entire line.

In Albritton and McMullen [2007], authors propose an Ant Colony Optimization (ACO) for the product design problem. Results show that ACO is a suitable method for problem instances with similar sizes. However, the method uses many input parameters and solutions can be dependent on the fine-tuning of these parameters. As our method ought to be general, this fine-tuning is not desired.

3 Problem description

Blauw Research provides companies with a tool that is able to simulate the market for their products. Now they also want to implement an optimizer into the tool. The methodology of simulating the market lies outside the scope of this research, but we will briefly address this in this section for completeness. We start by giving our definition of the product design and its complexities for our problem. Following, we explain how conjoint analysis is used in estimating customer preference. Then, a brief description of the market simulator is given and we conclude this section with an overview of what we want the optimizer to achieve.

3.1 Product design

A product is most commonly seen as a tangible object like beers, hairdryers or cellphones. However, we also consider products as phone subscriptions and bank accounts. The products just mentioned have one similarity that is important for this research: the markets on which they exists are oligopolies. The majority of such markets is provided by only a small number of competitors. In addition, we only focus on the oligopolies with heterogeneous product as optimizing product design in homogeneous markets would not be very interesting.

We assume that all products in the market have a similar structure. This means that they consist of the same attributes, but the level of the attribute that is chosen can differ between products. If we take a hairdryer for example, an attribute would be the wattage (and the drying power that comes with it) and the level of the wattage. Each hairdryer does have drying power, but one could have stronger drying power than the other. The same holds for bank accounts, all banks provide an interest rate but the rates can be different. Attributes can have discrete levels like certain design features, attachments or the brand. Continuous attributes are also possible, but usually this only concerns the price and in most cases this can also be discretized.

Furthermore, there can exist restrictions on the product design. Let us take the hairdryers as example again. A stronger, larger engine does not fit in a small design type or having pre-set drying modes requires the dryer to have an electronic user interface. Also, for some attributes their level can be fixed for certain products. Mainly brand is such an attribute, it is usually one the most important features of a product but it is fixed to the company that offers that product. It can also be the case that a company advertises their products as being the cheapest and therefore requires that the price is lower than any of the competitors.

Finally, attribute levels can also be associated with different costs. We only consider variable costs for the different levels as we do not have any other data, we are therefore only able to calculate market share, revenue and margins of the products. In the following section, we discuss how we are able to establish utilities to calculate these results given a certain product design.

3.2 Conjoint analysis

Blauw Research provides clients with market researches and one of the options that is offered is a conjoint analysis. With this, the way customer preferences work can be approximated so companies are able to make more substantiated decisions on their product design. For the conjoint analysis, clients provide Blauw Research with the set of all possible attributes and levels. Then, a survey is created where respondents repeatedly are presented with a set of products and each time have to state their favourable one. These sets of products are chosen such that within the set, products differ slightly from each other. After a certain number of iterations of this process, dependent on among other things the number of attributes and levels, we can estimate for each level of each attribute an utility value, which differs also for each respondent. For (near) continuous variables, the utility values are estimated on only a number of points and all other values can be calculated using interpolation. The points on which they are estimated are the same for all respondents and can be either given as input or left to decide by the program that calculates these utilities. The program used for calculation is the Lighthouse Studio provided by Sawtooth Software.

3.3 Market simulator

After having estimated the utilities, we have to decide on a method to transform utilities into decisions of the customers. We first calculate the total utility of a product, thus for all attributes we aggregate the utility per customer of its selected level. This is done for all products included in the simulation. We are now already able to use one possible method, which we refer to as the First Choice Method (FCM). This method boils down to simply having customers buy the product that is their first choice, thus the product for which their utility is highest. This method is incorporated in the market simulator. The concept is easy to grasp and can be computationally efficient. If our goal is to only change one product and keep remaining products fixed, we are only required to calculate the utilities of the remaining products first. Then we select for each customer the product that has, among these products, the highest utility. That product is for the given customer the status quo product. If we change one product, we recalculate its utility per customer and check whether it is higher than the status quo product for that customer. Thus if many recalculations are needed, this is an efficient method.

However, FCM also has its downsides. Estimations can be very poor if utilities are close to each other. Customers will select products with highest utility although the differences might be insignificant such that the customer would actually be indifferent. In the most extreme case, this method could indicate that a certain product would never sell when in fact it supplies half of the entire demand. To capture this, another customer decision method can be used: the Logit Choice Method (LCM). Here, one follows the same steps of the FCM. However, we continue by transforming the utility per product and per customer and then calculating the probability that a customer will buy a certain product. Let u_{ij} be the utility of customer i for product j and respectively $\mathcal{I}$ and $\mathcal{J}$ the sets of customers and products. We can then calculate the probability of buying a product as follows:

$$q_{ij} = \frac{e^{u_{ij}}}{\sum\limits_{k \in \mathcal{J}} e^{u_{ik}}} \qquad\qquad \forall\ i \in \mathcal{I}, j \in \mathcal{J}. \tag{1}$$

Here, q_{ij} is the probability that customer i will buy product j. The set of customers is represented by respondents of surveys. Thus to get the expected market share of product j given V, the total market value in number of customers, we perform following calculations.

$$M_j = \frac{\sum\limits_{i \in \mathcal{I}} q_{ij}}{\sum\limits_{i \in \mathcal{I}} \sum\limits_{k \in \mathcal{J}} q_{ik}} V \qquad \forall\ j \in \mathcal{J}. \tag{2}$$

The same holds for calculation of the market share M_j using FCM, but then probabilities q_{ij} are 1 if customer i buys product j and 0 otherwise. Calculations of revenue and margin are then respectively multiplying M_j with the price and the price minus costs of the corresponding product j. We use these same calculations for objective values in the product design optimization, which we will discuss next.

3.4 Optimizer

We will now elaborate on the optimization problem and certain features that increase the difficulty of the problem. Clients usually resort to market research when they introduce a new product to the market and want to know how to compile the product, but also to improve existing products. In both cases, it is almost always the case that these are not their only products in the same market. Changes of the products thus affect performance of their other products. Therefore, in optimizing either market share, revenue or profit, we have to take this into account. This increases the complexity of the problem, which is shown to be NP-hard by Kohli and Krishnamurti [1989].

In addition, we include the option of incorporating costs in the model which means there is also a trade-off between higher utility and higher costs that has to be made for all levels of all attributes. Many proposed heuristics use a local search component that maximizes utility. When including costs, these solutions can correspond to high costs and poor solutions. We also add the possibility of imposing constraints. This can be restrictions on certain attribute levels such as explained in Section 3.1, but also constraints to ensure that for example the market share of a product is above a certain threshold while maximizing its margin. These constraints again affect the complexity and performances of solution algorithms.

The size of some instances can raise issues as well. The total number of different product designs is calculated by multiplying the number of levels for each attribute and thus increases exponentially with the number of attributes. Other important factors on computation time are the number of respondents and if the LCM is used for modelling customer decisions also the number of products used in the simulation. These complicate the evaluation of functions (1) and (2). To find solutions relatively fast, we want an optimization method that is able to reduce the feasible region and requires only few evaluations of the objective value.

In many papers on product optimization, authors focus on the optimization of an entire product line (Belloni et al. [2008]). This is obviously a harder problem than only focusing on a single product. However, it usually is not applicable to real life problems. Developing and introducing a new product takes a long time and is associated with high costs. Therefore, clients commonly ask Blauw Research to investigate the "killing" of one product and replacing it with a new product. In addition, many of the problem instances are of such large scale that optimizing an entire product line takes too much computation time. The focus will thus only be on single product optimization. We now give a general mathematical programming formulation of this problem:

$$\max_x \sum_{j \in \mathcal{P}} M_j (p_j - c_j) \qquad , \tag{3}$$

$$s.t.$$

$$M_j = \frac{\sum\limits_{i \in \mathcal{I}} q_{ij}}{\sum\limits_{i \in \mathcal{I}} \sum\limits_{k \in \mathcal{J}} q_{ik}} V \qquad \forall\, j \in \mathcal{J}, \tag{4}$$

$$q_{ij} = \frac{e^{u_{ij}}}{\sum\limits_{k \in \mathcal{J}} e^{u_{ik}}} \qquad \forall\, i \in \mathcal{I}, j \in \mathcal{J}, \tag{5}$$

$$u_{ij} = \sum_{a \in \mathcal{A}} \sum_{l \in \mathcal{L}(a)} v_{ial} \cdot x_{jal} \qquad \forall\, i \in \mathcal{I}, j \in \mathcal{J}, \tag{6}$$

$$x_{jal} = N_{jal} \qquad \forall\, j \in \mathcal{J} \setminus \widehat{j}, a \in \mathcal{A}, l \in \mathcal{L}(a), \tag{7}$$

$$\sum_{l \in \mathcal{L}(a)} x_{\widehat{j}al} = 1 \qquad \forall\, a \in \mathcal{A}, \tag{8}$$

$$c_j = \sum_{a \in \mathcal{A}} \sum_{l \in \mathcal{L}(a)} k_{al} \cdot x_{jal} \qquad \forall\, j \in \mathcal{J}, \tag{9}$$

$$p_j = \sum_{l \in \mathcal{L}(a^P)} x_{ja^P l} \cdot b_l \qquad \forall\, j \in \mathcal{J}, \tag{10}$$

$$x_{jal} \in \{0, 1\} \qquad \forall\, j \in \mathcal{J}, a \in \mathcal{A}, l \in \mathcal{L}(a). \tag{11}$$

The formulation can be interpreted as follows:

- The objective value (3) is the margin of all products in the clients portfolio $\mathcal{P}$. This is calculated as the market share M_j of each product $j \in \mathcal{P}$ times the difference between the corresponding price p_j and costs c_j. When costs are set to zero, the objective value becomes revenue and if in addition the price is 1, market share is maximized.

- Constraints (4) and (5) are used for the calculation of the market share M_j and are already explained via equations (2) and (1), respectively. Notice that we chose here for the LCM method for customer decisions due to downsides of FCM that we discussed before.

- Utilities used in (5) can be acquired via constraint (6). v_{ial} contains the utilities of customer i for the level l of attribute a. As x_{jal} is equal to 1 if level l is selected for attribute a for product j, the right hand side of the equation denotes the utility for all products $j \in \mathcal{J}$.

- Restrictions (7) fix all products, except $\widehat{j}$, to the null scenario. Here, $\widehat{j}$ denotes the product that is optimized.

- To ensure that also for product $\widehat{j}$ a complete product is formed, we imply constraints (8) and (11). Then exactly one level is selected for each attribute. Note here that continuous attributes are discretized for reasons which are discussed in Section 4.

- The costs associated with product j (c_j) are calculated using restrictions (9). Costs of level l for attribute a are represented by k_{al}. If x_{jal} is equal to 1, the corresponding costs are added to the product.

- Constraints (10) are used to link the price to the products j. Attribute a^P represents the price attribute, thus x_{ja^Pl} is 1 if price level l is selected for product j. Then b_l is the corresponding price for that product.

As stated, this is only a general formulation. In many problem instances, additional constraints are imposed. Any type of constraint can be incorporated in this formulation.

4 Data analysis

For testing our methodology, we will focus on an example of the hairdryers markets of China. A null scenario, corresponding to the current state of the market, was delivered to Blauw Research and a conjoint analysis was performed to estimate utilities. In this section, we will present an overview and analysis of the hairdryer case.

The data was gathered by Blauw Research using a conjoint analysis. A survey was set out in China and was answered by 1038 respondents. The Chinese hairdryer market is represented by 21 hairdryers divided over 6 customers. Of these products, 8 belong to the client for which the optimization is performed. A hairdryer is assumed to possess 15 attributes that can influence a customers choice decision. Also, each attribute has between 3 and 12 levels and for price we can even choose from over 400 different levels. This is a relatively large case as without any constraints, there are more than 450 billion possible designs for a hairdryer. However, part of these designs are actually infeasible due to imposed restrictions.

We now present some interesting findings on the utilities. As stated in Section 3, we aim for a general optimizer. Therefore, irregularities in data are not used in the development of methods. It is however interesting for conclusions on results. The conjoint analysis for this case was quite unusual. One of the attributes incorporated is the appearance of the hairdryer, for which 12 different options are possible. These 12 options consist of 6 different models which are all offered in either black or white. Beforehand, it was decided which respondents would be shown black hairdryers or white hairdryers. The research was therefore basically split up into two different researches for each colour. Then, utilities were also estimated separately for both colours, but afterwards the data was concatenated. The 6 appearance options of both colours were thus combined into 12 options in total. For the models of the colour that was not shown to a respondent, those utilities were set to -100. Whereas utilities normally add up to 0, for the appearance attribute this is not the case. In addition, estimating utilities of other attributes is affected by which survey is taken by a customer.

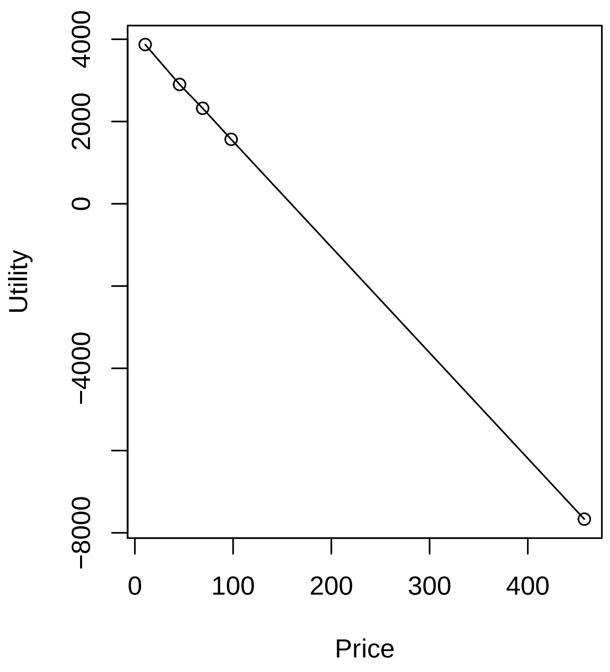

Figure 1: Aggregated utilities for price

Price is in almost all cases an important factor for customer decisions. Higher prices are accompanied with lower utilities. Usually, this relation is non-linear as utilities tend to decrease harder for higher prices. We plotted the utilities for price, summed over all respondents, in Figure 1. As can be seen for this case, the relation between price and utility is practically linear. The points in the graph denote the points on which utility was estimated. The first points are all close to each other whereas the difference with the last is relatively large. In this gap, true utilities might show a more non-linear relation. This is not captured as we use a linear interpolation for estimating intermediate points.

Furthermore, we notice a lot of indifference among respondents. More specifically, 45% of all utilities are zero. These zeros always occur for all levels of a specific attribute. This occurs if a respondent has replied in the survey that that attribute is irrelevant. This indifference can be found for all attributes except price. The percentage of respondents that have zero utilites for an attribute varies from 15% to even 90% for some attributes.

5 Methodology

In Section 3 we derived some guidelines that our optimization model has to conform to. We will use the Nested Partition (NP) method as described in Shi et al. [2001] as the basis of our optimization method. However, we will apply a few required adjustments to the method since our problem is more complicated and larger than instances used in Shi et al. [2001]. First, we will give an outline of the NP method and our implementation. Then, respectively, we discuss how the algorithm is extended by Genetic Algorithms (GA) and a CTREE method, which is a mixture of Greedy Search (GS) and randomness. GA and GS are also included in Shi et al. [2001] but, especially GS, require modifications to cope with our problem instances.

Table 1: Attribute levels for hairdryer example

Colour	Drying Power	Price
Black	High	49.99
White	Medium	99.99
	Low	

In the following sections, we will make use of some visualizations to clarify the methodology. We again resort to hairdryers, but use a simplified example for illustration. The possible designs can be created by any combination of levels for the 3 attributes given in Table 1. Also, some results on margin are presented in Sections 5.2 and 5.3. These are fictional and solely used for clarifications.

5.1 Nested Partition

The NP method is useful for optimization of product designs as it is able to split feasible regions and focus only on the promising parts. For product design, the feasible region is initially represented by all possible product profiles. Splitting up this feasible region is done iteratively per attribute. Let us illustrate this with our hairdryer example via Figure 2. The nodes represent feasible regions with the size of the region between brackets. At the start of the algorithm, the hairdryer is an empty product. As can be seen in the root node, the feasible region then consists of 12 hairdryers.

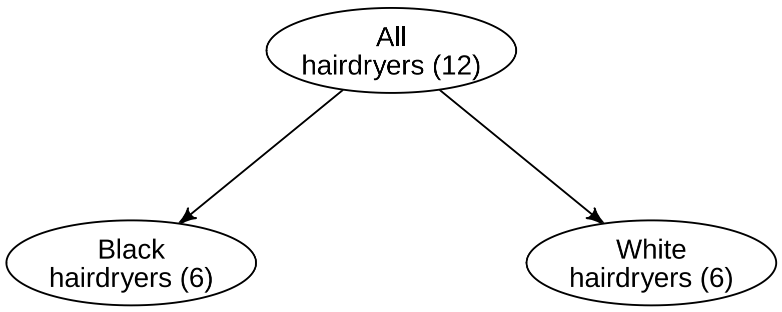

Figure 2: Nested Partition at depth 0

We continue by deciding whether the hairdryer is going to be black or white. The feasible region is then split up into all possible black hairdryers and all possible white hairdryers. So if we are now able to decide in which subregion the optimal solution can be found, we reduced the size of the feasible region by 50%. If for example the optimal hairdryer is black, we move to this subregion and obtain Figure 3. For this new reduced feasible region, we can repeat this procedure for drying power and price. After selecting the price, the feasible region is singular and we are left with a complete hairdryer. This hairdryer would then be the global optimal solution.

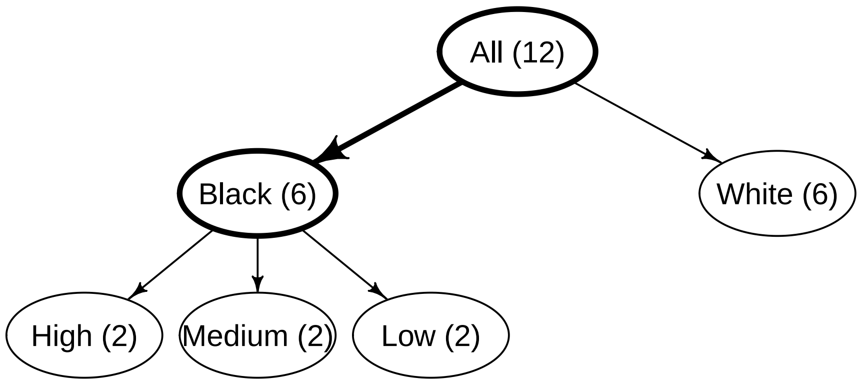

Figure 3: Nested Partition at depth 1

Unfortunately we are, in a reasonable amount of time, not able to state with certainty whether a subregion contains the optimal solution. Instead we estimate how promising a subregion (or attribute level) is and reduce the feasible region to the most promising subregion. This would again be repeated until a complete product is established. The product design found is then a local optimal solution. The probability that also the global optimum is found, depends on the method of estimating a subregions promisingness.

As we know from Section 3.4, evaluations of objective values can be computationally expensive. To decide how promising a subregion is, we prefer methods with only few evaluations. In addition, we only have a part of the product. Evaluating this does not give full insight in the profitability of the product. To solve both these issues, we randomly sample the remainder of the product. Objective values of this product can be calculated and with these values could be decided how promising a region is. If the majority of a product is randomly formed, which is the case early on in the algorithm, the probability of having a poor product while it was selected from an actual very promising region is quite high. To prevent this from happening, we randomly draw multiple products from each subregion. We then evaluate all these products and take the highest objective value per region as index of how promising that region is. In deciding on the number of products to select per region, one has to make a trade-off between computation times of the algorithm and in probability of selecting the actual most promising region.

Since computation times must be reasonable, we will have in many cases a positive probability of selecting a region that does not contain the global optimum. If we implement the method as described until now, we have no chance of finding the global optimum after choosing the wrong region. Therefore, the idea of backtracking is included in the NP algorithm. Next to estimating the promisingness of the subregion, we also do this for the super region. In Figure 4, we currently have selected black hairdryers with high drying power. The two promising regions to investigate are a price of 49.99 (PR 1) and 99.99 (PR 2). In addition, we also sample solutions from the aggregate region OR. Similar to the promising regions, we can sample any number of solutions here which should be calibrated based on results. If only one solution is sampled, this could for example be a white hairdryer with medium drying power and a price of 99.99. Let us assume this hairdryer has a higher objective value than the hairdryers of the promising regions PR 1 and PR 2. We now have to backtrack to a super region. This is done by removing the most recently added attribute and for that region, the subregions and super region are again investigated. In Figure 4, this would mean removing the high drying power which results in returning to the situation of Figure 3.

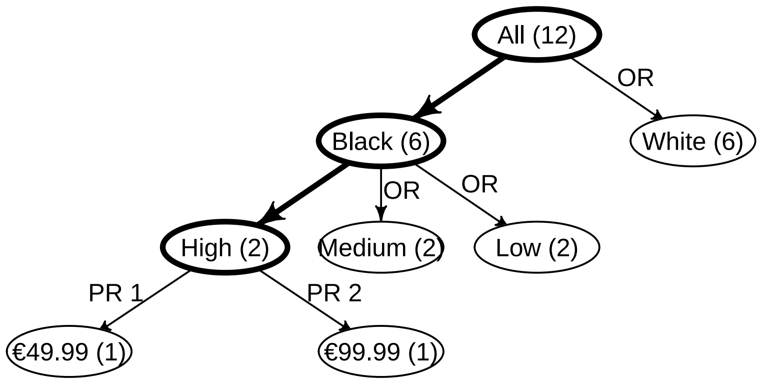

Figure 4: Nested Partition at depth 2

The NP algorithm we just described runs for a given amount of computation time. For reasons we will describe in Section 5.2, we keep track of our randomly sampled solutions. Therefore at search termination, we are able to return the solution with the highest objective value found up to that point. Completing the algorithm, so reaching a singular most promising region and a complete product, does not terminate it. We then restart the algorithm but randomize the order of attributes, except for the price which will always be the last attribute. Price is selected last such that given a certain product design, revenue and margin can be optimal. By changing the order, the subregions are different so the algorithm ends less in a certain local optimum. Since the algorithm uses much random sampling, finding multiple local optima is likely. However, this decreases by including a Greedy Search into the NP method which we discuss in the upcoming section.

5.2 CTREE

Similar to Section 5.1, we give an entire description of the algorithm to be able to clearly describe our own adjustments. The CTREE is included to improve estimations of the promising region index. In the base version of the NP algorithm, we complete partial products by randomly selecting levels for remaining attributes. As alternative, Shi et al. [2001] propose a Greedy Search (GS). Then, not every attribute is determined randomly but based on the maximization of utility. For a certain attribute a, GS is performed with a given probability p_a and an attribute level is randomly selected with probability $1-p_a$.

The attribute level selection using GS is quite simple, the level with the highest utility summed over all respondents is selected. Let us illustrate this with Table 2. In this example, we represent the market by three respondents. For each of these respondents, the utility is estimated for the levels of in this case the attribute "Drying Power". In the bottom row, the summed utility is given. Thus if GS is used in the selection of this attribute, a medium drying power would be selected. This method is based on problem instances that differ from ours and is therefore not suitable in this manner. For example, Shi et al. [2001] do not include costs and the idea of appending cannibalization to the objective function is proposed, although not carried out.

Table 2: Example utilities for drying power

Respondent	High	Medium	Low
1	3	0	-3
2	-2	1	1
3	-1	0	1
Total	0	1	-1

Both costs and cannibalization can occur in our problem. In the case of costs, greedily selecting attribute levels with highest utility can lead to excessive costs such that less preferred levels are more profitable. It is reasonable to state that medium drying power costs more to establish such that in our example of Table 2 the low drying power would result into higher margins. Furthermore, as utilities differ for respondents, it can be profitable to offer different products to reach as many customers as possible. As can be seen in Table 2, we can appeal more customers by offering two products. One with high drying power that is favoured by the first respondent and one with low drying power that is preferred by the other respondents.

Products will be composed to do so if we include cannibalization in our objective function. In this case, we do not want to focus on attribute levels that has highest summed utility over all respondents but appeal customers that do not yet purchase any of our products. A greedy search is then again not suitable. However, selecting attribute levels in a more sophisticated way than randomly is a good idea to improve the promising region estimation. So instead of performing GS based on utilities, we develop a method of estimating which levels have highest impact on the objective value.

To do so, we adopt machine learning into our algorithm. In selecting the most promising region, we calculate the objective values of multiple products. We now keep track of all these solutions and their objective values. With this information, we can estimate regression trees. Such trees are easily interpreted and user friendly for making predictions. However, they are applicable in a different manner for our problem. Usually, one reads the tree by starting at the root node and following the branches according to the data they want to predict or explain and find according values at the terminal nodes. We start at the terminal node with highest average value and then find the corresponding variable splits. We illustrate this with a tree for the drying power attribute in Figure 5. Here, the left node is the terminal node with the highest average margin of slightly below 900. The high and low drying power hairdryers show significantly more margin than the hairdryers with medium power. Now, with probability p_a^{CTREE} the random sampling is restricted to hairdryers with high or low drying power. With probability $1 - p_a^{CTREE}$, the drying power is selected completely random. In the case of Figure 2, this procedure is first performed for the drying power attribute and then for the price to create a single hairdryer in a promising region. Again, multiple hairdryers are created per promising region and the highest objective value found is taken as promising region index.

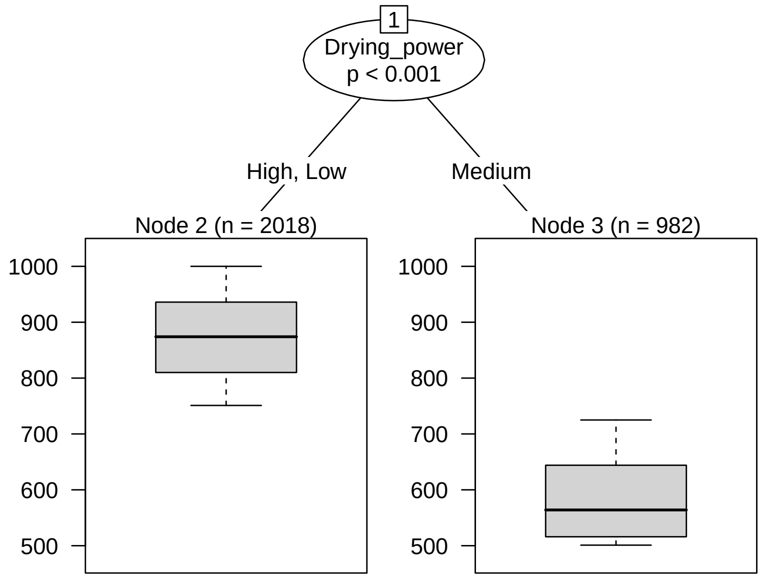

Figure 5: Regression tree of margin

This new method is thus a mixture of greedy and random and we will refer to it as CTREE. Next to being able to use the concept of GS again, it also suits the NP framework better. Early on in the algorithm, we have found only few solutions. The diversification is then quite high in the random sampling as the regression tree is not likely to have any splits. The sampling is then completely random. The more solutions we gather, intensification increases as the tree is able to make more accurate estimations. In addition, we set a maximum number of solutions that are stored. If this cap is reached and a new solution is found, we add this solution if its objective value is higher than the lowest objective value in the set. The solution with lowest objective value is then removed. Note that a solution is never added if its duplicate is already present in the data set.

For establishing trees such as in Figure 5, we adapt the conditional inference framework as described in Hothorn et al. [2006]. These obtained trees overcome some well known problems in comparable regression tools. The first is overfitting. Overfitting would in our case be an issue as it decreases diversification. The regression tree would try to fit the current data set too much and especially if only few solutions are found, we might be focusing on regions that are actually not the most promising. The second problem is the bias to variables with many possible splits. Since attributes can be both continuous and discrete, the algorithm would focus less on promising levels of discrete attributes and might thus lack some important intensification. Additionally, the conditional inference trees are able to capture multiple variable types, such as ordinal, nominal or numeric. Implementing the trees is done with the *partykit* package in R, which is a reimplementation of the *party* package provided by Hothorn et al. [2006].

As stated in Section 5.1, we randomize the order of attributes each time the NP algorithm repeats itself. The first time we determine the order differently. We then perform random mutations with a given probability p_a^{IO} for each attribute in the null scenario of the product that is to be optimized. We create a specified number of these mutated products and calculate objective values. We then estimate the regression trees for each attribute. The attributes are then ordered based on their highest terminal node mean value, which in Figure 5 would be approximately 875. The attribute for which that value is the highest is first in the order and so on, except for the price which will again be last in the order.

5.3 Genetic Algorithm

The next extension on the NP algorithm is the Genetic Algorithm (GA). Like GS, the GA is incorporated to improve the selection of the promising region. After having sampled a certain number n products from a subregion, the GA is executed. From these products, the best $\frac{n}{2}$ are selected for crossover but will also be added to the new solution set in their current form. In the crossover, we randomly create $\frac{n}{4}$ couples. For each couple, their levels for attributes are swapped with a probability p_a^{GA} per attribute.

Table 3: Initial solution set

	Design 1	Design 2	Design 3	Design 4
Colour	Black	Black	Black	Black
Drying power	Medium	Medium	High	Low
Price	49.99	49.99	99.99	49.99
Margin	520	520	825	730

Let us clarify this again by visualization. In Table 3 we show four solutions that could be obtained during the investigation of the promising region of black hairdryers. To recall, this means we are in the situation of Figure 2. As we have sampled four solutions, the

GA uses two solutions for crossovers. The two best solutions are Design 3 and Design 4 and as there are only two, they will form a couple. Then for each attribute we swap the levels within the couple. If for example Drying power will be swapped and other attributes remain constant, the initial solution set of Table 3 is transformed into the solution set of Table 4.

Table 4: Solution set after a GA iteration

	Design 3	Design 4	Design 3'	Design 4'
Colour	Black	Black	Black	Black
Drying power	High	Low	Low	High
Price	99.99	49.99	99.99	49.99
Margin	825	730	685	900

The first two solutions, Design 3 and Design 4, of Table 4 are the two initial solutions used for crossovers and the other two are the ones obtained by crossover. Contrary to Shi et al. [2001], we do not include mutation in the GA as we want to conserve solutions. Mutation is usually included to avoid local optima (Grefenstette [1986]). However, we want to estimate how promising a certain subregion is and this is evaluated by the highest objective value found. Therefore, in this application of the GA, intensification is more important than diversification.

6 Computational experiments

We will now test above presented methods by means of numerical experiments. As stated, we focus on the hairdryer case. As given in our data analysis in Section 4, this is a relatively large instance for the product design problem. Consequently, we were not able to find solutions that guarantee optimality. We therefore focus on results relative to the null scenario. Now, we present the test environment.

6.1 Test environment

For this problem, we focus on the optimization of margin. With our method, we are able to optimize the margin of all products by adjusting a single product. There are 8 products in the clients portfolio. We optimize each one of them keeping the others equal to the null scenario so after optimizing, only one product is changed. We can then state which product should be killed and replaced with another to have the highest increase in margin. Also, this provides us with more insight in the performance of the optimizer given different starting solutions.

Table 5: Parameter values

Name	Description	Value
PRS	Number of solutions sampled from each promising region	40
ORS	Number of solutions sampled from the super region	40
CT	Computation time of the algorithm in minutes	60
MS	Maximum number of solutions stored for CTREE estimations	500
N^{IO}	Number of solutions created to determine initial order	200
p_a^{IO}	Probability of mutating attribute a to create the N^{IO} solutions	0.75
p_a^{CTREE}	Probability of using CTREE instead of randomly sampling for attribute a	0.5
p_a^{GA}	Probability of swapping the levels of attribute a of a pair during the GA stage	0.25

Unless stated otherwise, the algorithm is executed with parameter values given in Table 5. We will compute results for multiple versions of the NP algorithm. Our main method will be the NP algorithm in combination with GA and CTREE, from now on referred to as NP GA CTREE. As we noted that a Greedy Search is not compatible with this problem, the closest comparison we have to the methodology of Shi et al. [2001] is the NP algorithm with only GA. Let this be abbreviated to NP GA. Thus, our first evaluation will be of the inclusion of CTREE. Furthermore, we compare the randomization of the attribute order to having a fixed order.

6.2 Results

We now present results using the test environment discussed in Section 6.1. In Table 6, results of our main method NP GA CTREE are given. In the first column, we present which product is optimized. The remaining columns show respectively the market share, revenue and margin of all products achieved after the optimization of margin. Results are an average over 5 runs of the algorithm where the value between brackets represent the standard deviations in percentages.

Table 6: Results after optimizations of margin using NP GA CTREE

Optimized product	Market share (%)	Revenue (€)	Margin (€)
Null scenario	68.60	46041.39	-3166.58
1	$70.49_{(0.67\%)}$	$78173.77_{(4.49\%)}$	$28236.53_{(9.23\%)}$
2	$71.32_{(1.37\%)}$	$80717.87_{(7.74\%)}$	$25120.66_{(11.62\%)}$
3	$71.52_{(0.34\%)}$	$78396.10_{(2.62\%)}$	$21829.30_{(5.66\%)}$
4	$69.02_{(0.41\%)}$	$83032.68_{(2.58\%)}$	$25152.69_{(7.76\%)}$
5	$70.94_{(0.28\%)}$	$85787.60_{(2.99\%)}$	$27956.14_{(4.05\%)}$
6	$69.41_{(0.65\%)}$	$82532.95_{(4.93\%)}$	$25732.12_{(4.69\%)}$
7	$70.27_{(0.44\%)}$	$83275.84_{(4.18\%)}$	$25669.08_{(7.44\%)}$
8	$69.82_{(0.47\%)}$	$85759.65_{(3.12\%)}$	$26961.65_{(5.45\%)}$

The most striking result is the huge increase in margin that is obtained. Whereas there was a loss in the null scenario, the optimization of any product turns this into a positive margin of approximately 7 to 10 times the absolute value of this loss. This could be due to all clients products being far from optimal in the null scenario. However, it might also be due to errors in utilities. Next to high increases in margin, the total revenue almost doubles. The market share also shows a slight improvement after optimization. The highest margins can be found for the optimization of products 1 and 5 whereas optimizing product 3 leads to a considerably lower margin than the other products. Notice that for product 1 the lowest average revenue is achieved whereas the average margin was highest. In the null scenario, costs of product 1 are more than twice the costs of the other products. Thus in optimization, high increases in margin can be obtained by cutting these costs.

In the optimization of the 8 different products, the designs of the optimized hairdryers were quite similar to each other. Independent of the original appearance, optimization almost always led to selecting the first white appearance option. In the null scenario, products 1, 5 and 8 already have this appearance. As can be seen in Table 6, these products also perform best in optimization. Furthermore, the price was always set to the highest level. This is a peculiar outcome as in the null scenario none of the hairdryers have a price equal to the maximum. A possible explanation would be that higher prices do not lead to realistic drops in utility. This might also cause the high increases in revenue and margin.

Table 7: Results after optimizations of margin using NP GA

Optimized product	Market share (%)	Revenue (€)	Margin (€)
Null scenario	68.60	46041.39	-3166.58
1	$70.38_{(0.98\%)}$	$78177.29_{(6.63\%)}$	$27826.51_{(6.70\%)}$
2	$71.50_{(0.63\%)}$	$83540.24_{(3.17\%)}$	$25161.43_{(8.81\%)}$
3	$71.85_{(0.72\%)}$	$82037.07_{(3.86\%)}$	$23480.36_{(8.62\%)}$
4	$69.08_{(0.61\%)}$	$83459.10_{(4.55\%)}$	$26669.27_{(5.42\%)}$
5	$70.98_{(0.48\%)}$	$86857.62_{(3.56\%)}$	$28158.43_{(4.30\%)}$
6	$69.43_{(0.49\%)}$	$81470.58_{(4.73\%)}$	$25140.97_{(8.99\%)}$
7	$70.43_{(0.58\%)}$	$82286.91_{(2.94\%)}$	$25000.59_{(9.41\%)}$
8	$69.85_{(0.92\%)}$	$87609.48_{(6.45\%)}$	$27151.02_{(11.03\%)}$

We now discuss results of NP GA for evaluating our CTREE extension. These results can be found in Table 7. The structure of the table is similar to Table 6 and again 5 runs are performed. We notice that results are very similar to those from our NP GA CTREE method. This can be seen in Figure 6, where optimized margin for each product is plotted for all three methods. As before, products 1 and 5 show the highest increase in margin with product 8 ranking third again. Products 3 and 4 show better performance using NP GA while 6 and 7 result in higher margin when including CTREE. Notice however that differences between methods are rather low in comparison with the standard deviations. Thus, both NP GA CTREE and NP GA are suitable methods to optimize product designs of which it is not possible to show if one outperforms the other.

Table 8: Results after optimizations of margin using NP GA CTREE with fixed attribute order

Optimized product	Market share (%)	Revenue (€)	Margin (€)
Null scenario	68.60%	46041.39	-3166.58
1	$70.13_{(1.03\%)}$	$74912.10_{(6.68\%)}$	$25590.44_{(9.46\%)}$
2	$71.27_{(0.49\%)}$	$81977.77_{(3.93\%)}$	$24934.64_{(9.90\%)}$
3	$71.62_{(0.78\%)}$	$80643.05_{(4.32\%)}$	$23605.57_{(3.94\%)}$
4	$68.83_{(0.74\%)}$	$82293.41_{(5.23\%)}$	$25000.06_{(7.68\%)}$
5	$70.73_{(0.92\%)}$	$84096.38_{(4.08\%)}$	$26615.95_{(9.92\%)}$
6	$69.20_{(0.77\%)}$	$80326.82_{(5.34\%)}$	$24878.02_{(7.90\%)}$
7	$69.78_{(0.82\%)}$	$76845.57_{(6.30\%)}$	$21912.21_{(12.19\%)}$
8	$69.45_{(0.79\%)}$	$83523.41_{(2.36\%)}$	$25419.65_{(8.29\%)}$

In Table 8, again similar to the previous two tables, we present results of the NP GA CTREE method when we keep a fixed order of selecting the attribute levels. Here again, optimizing products 1, 5 and 8 results in the highest margin. However, the margins are considerably lower than for the methods with randomized attribute order, except for product 3 for which average margin is higher than for both other methods. Figure 6 clearly shows that having a fixed order is outperformed by randomized orders. Note that again standard deviations are quite high and results are based on only 5 runs.

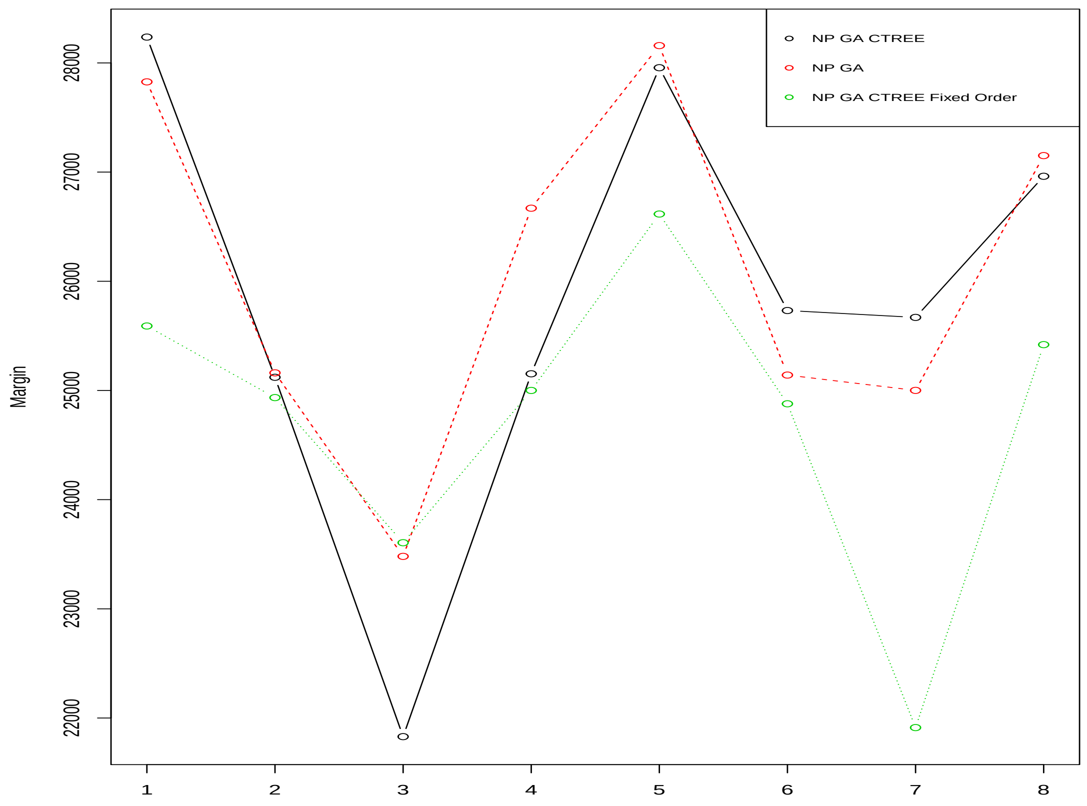

Figure 6: Comparisons of margins from Table 6, 7 and 8

6.3 Sensitivity Analysis

Now we check how robust the solutions of our optimization are. This is done by making perturbations on the utilities. Here, we focus on two different parts. First, we make adjustments to the utilities of price. As stated in section 6, the price was set to the highest possible level in all optimizations. This might be due to the aggregated utility being practically linear (Figure 1). Drops in utility due to higher prices does not impact the total utility sufficiently. Our second analysis will focus on overall robustness and the difference between black and white hairdryers as in almost all optimizations white hairdryers were formed. As explained in Section 4, respondents currently have a strict preference for one colour. However, no reasons are given that this is a realistic representation and therefore white hairdryers might not perform as well as they do now.

For the analysis, we selected the solution with highest margin over all runs of all methods. This solution was obtained by optimizing product 1 using the NP GA CTREE method, with randomized order of attributes. Results of this optimization can be found in Table 9 in the row "Initial". The remaining rows represent results if utility is adjusted. Hereby, we subtract a certain percentage of the absolute value of utility of the last point estimation in Figure 1. Let this point estimation for a respondent i be u_i^{PE}. The new utility $\hat{u_i^{PE}}$ can then be calculated as follows:

$$\hat{u}_i^{PE} = u_i^{PE} - p^{decrease} \times |u_i^{PE}|$$

Here, $p^{decrease}$ is the percentage in the first column of Table 9 divided by 100. Note that this also affects interpolated utilities between the fourth and the final point estimates, resulting in a more concave utility function for price. It can be seen that the 5% decrease leads to a higher margin but a huge decrease in market share. The margin again decreases for 10% and 20% decreases but shows the highest margin if 50% decrease is applied. Note that the margin is aggregated over all clients' products of which prices vary widely. This illustrates that changes in utility of price have a high impact, but that the problem is non-convex.

Table 9: Sensitivity Analysis on price

$p^{decrease}$	Market share (%)	Revenue (€)	Margin (€)
Initial	70.59	79980.58	31884.10
5% decrease	44.12	67666.40	33719.7
10% decrease	44.07	65661.26	32114.73
20% decrease	43.90	60279.13	27809.43
50% decrease	44.19	69825.28	35445.97

We now assess the overall robustness of the solutions. Random perturbations will be made on the utility values in the data set to evaluate how this affects margin. As stated in Section 4, utilities of white or black appearances were set to -100 if these were not shown to the respondents. Since this has a huge impact, we first make adjustments here. For every appearance with utility -100, the appearance of the other color has a regular utility. The utility of -100 will then be set equal to the regular utility as if customers have no preference in colour. We continue with the same solution used in the previous sensitivity analysis. As utilities have already changed, we obtain different results which can be found in Table 10 in the row "Initial*". In the columns, for all objective values the minimum, maximum and mean over the runs are given. In the columns for the mean, the standard deviations in percentages are given between brackets. As "Initial*" does not change over runs, these values are the same for all three statistics.

The remaining rows represent results over 100 samples of perturbations. We randomly select 1%, 5% and 10% of all possible data points, which is the number of respondents times the number of levels of all attributes. Let u_{al} be the utility of such an attribute level. The perturbated utility $\hat{u_{al}}$ is then as follows:

$$\hat{u}_{al} = u_{al} + \sigma_{al} \times z$$

Here, σ_{al} is the standard deviation of the utilities of all respondents for attribute level al and z is a random number drawn from the standard normal distribution.

Table 10: Sensitivity analysis on utilities

	Market share (%)			Revenue (€)			Margin (€)		
	Minimum	Mean	Maximum	Minimum	Mean	Maximum	Minimum	Mean	Maximum
Initial		70.59			79980.58			31884.10	
Initial*		52.03			58266.78			29198.32	
1% perturbations	51.16	51.76$_{(0.48\%)}$	52.36	54789.15	57635.51$_{(1.54\%)}$	59350.55	26560.04	28656.30$_{(2.21\%)}$	29848.31
5% perturbations	49.27	50.97$_{(1.00\%)}$	52.54	51892.73	56079.95$_{(2.99\%)}$	59824.07	23886.47	27182.97$_{(4.32\%)}$	29548.85
10% perturbations	48.13	50.00$_{(1.22\%)}$	51.28	48339.08	53846.91$_{(3.24\%)}$	57200.72	21103.32	25121.98$_{(5.06\%)}$	27758.40

We first notice that the assumption of indifference between colours leads to a slight decrease in margin and a huge drop in revenue and market share. For perturbations, a clear relation is seen between the number of perturbations and the performance. All statistics for market share, revenue and margin show a decrease if more changes are made in the utilities. Especially the lowest margin found when 10% of utilities are different deviates highly from the mean value of "Initial*" as it decreases by almost 30%.

7 Conclusion

In this research, we extended on methods for the optimization of product design. Using the NP method proposed by Shi et al. [2001] as a base model, we made adjustments to apply this method to the optimization of margin. The Greedy Search part was replaced by the CTREE method and a randomization of the attribute order was introduced to improve performance. We derived following conclusions on our research:

- All methods tend to work well. Optimizations lead to high increase in objective values compared to the null scenario. As there was no method of optimization before, our optimization model is a good addition to the market simulator.

- To improve the estimation of the promising region index, we included the CTREE method. Margins highly increased using this methodology. However, we were not not able to show that performance of NP GA CTREE was significantly better or worse than of NP GA.

- The methods NP GA and NP GA CTREE both randomized the attribute order after a complete product was formed. Results of NP GA CTREE were also shown if the order was fixed throughout the optimization. Margins were considerably higher for the optimization of nearly all products if randomization was included.

- The optimization of each different product led to one specific appearance option. Products that already possessed that option resulted in the highest increase in margin. Having more similar products in the product line decreases margin of the other products. However, it might also be a consequence of the data. The CTREE method uses a set of the best solutions found until then. Changing colour of the hairdryer has a huge impact on the objective value. It might therefore take longer computation time to find good hairdryer designs if the appearance option is initially not the optimal option.

- For all methods, standard deviations on margins after optimization are high. Also, the randomization of attribute order showed better results. Thus if the attribute order influences performance, increasing computation time and thus the number of different orders used should lead to higher margins.

- It was shown that perturbations of the utilities influence the margin after optimization. If the utilities of higher prices were decreased, the total margin increased in most cases. The loss of margin of the optimized product with highest price was compensated by other products in the product line. For the sake of robustness, it is thus important to vary prices over the product line. Furthermore, applying perturbations on parts of the entire utility set has a high impact on margin. In the worst case, margins even decreased by 30%. As solutions are sensitive, it is thus important to have accurate estimations of utilities.

We conclude with a few remarks on our research and possible further research on our methodology. First, let us emphasize that results and conclusions are based on 5 runs per method. To draw better and more justified conclusions, more iterations and longer optimization times are required. As 1 run consisted of 8 different optimizations taking each one hour, obtaining current results already takes a long time. Within the time framework, more extensive computational experiments were unfortunately not possible.

Furthermore, we were not able to assess performance of our methods compared to the global optimum. Due to the size of this instance, we were not able to find the optimal solution. For future research, it would be interesting to test our methodology on smaller instances and make comparisons with other heuristics.

Finally, we propose a rather simple method to improve robustness of the optimization. Our sensitivity analysis might not represent realistic utilities. However, if it is possible to make such perturbations, robustness could easily be incorporated in the objective function. The objective value is then not calculated for a single set of utilities, but averaged over multiple data set. It is then also possible to select solutions based on a level of robustness, such that in for example 95% of the cases a certain margin is achieved.

References

M. David Albritton and Patrick R. McMullen. Optimal product design using a colony of virtual ants. *European Journal of Operational Research*, 176(1):498–520, 2007.

P.V. Balakrishnan and V.S. Jacob. Genetic algorithms for product design. *Management Science*, 42(8):1093–1227, 1996.

Alexandre. Belloni, Robert. Freund, Matthew. Selove, and Duncan. Simester. Optimizing product line designs: Efficient methods and comparisons. *Management Science*, 59 (9):1544–1552, 2008.

Paul E. Green, J. Douglas Carroll, and Stephen M. Goldberg. A general approach to product design optimization via conjoint analysis. *Journal of Marketing*, 45(3):17–37, 1981.

Paul E. Green, Abba M. Krieger, and Yoram Wind. Thirty years of conjoint analysis: Reflections and prospects. *Interfaces*, 31(3):S56–S73, 2001.

John J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986.

Torsten. Hothorn, Kurt. Hornik, and Achim. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

J.R. Jiao, T.W. Simpson, and Z. Siddique. Product family design and platform-based product development: a state-of-the-art review. *Journal of intelligent Manufacturing*, 18(1):5–29, 2007.

Rajeev Kohli and Ramesh Krishnamurti. Optimal product design using conjoint analysis: Computational complexity and algorithms. *European Journal of Operational Research*, 40:186–195, 1989.

R. Duncan Luce and John W. Tukey. Simultaneous conjoint measurement: A new type of fundamental measurement. *Journal of Mathematical Psychology*, 1(1):1–27, 1964.

Leyuan Shi, Sigurdur Ólafsson, and Qun Chen. A new hybrid optimization algorithm. *Computers and Industrial Engineering*, 36(2):409–426, 1999.

Leyuan Shi, Sigurdur Ólafsson, and Qun Chen. An optimization framework for product design. *Management Science*, 47(12):1581–1732, 2001.

Leyuan Shi and Sigurdur Ólafsson. Nested partitions method for global optimization. *Operations research*, 48(3):390–407, 2000.