DATA TREND MINING FOR PREDICTIVE SYSTEMS DESIGN

BY

CONRAD SOORBA TUCKER

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Systems and Entrepreneurial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Doctoral Committee:

Assistant Professor Harrison M. Kim, Chair
Loretta Auvil
Professor Deborah Thurston
Professor Yuanhui Zhang

Abstract

The goal of this research is to propose a data mining based design framework that can be used to solve complex systems design problems in a timely and efficient manner, with the main focus being product family design problems. Traditional data acquisition techniques that have been employed in the product design community have relied primarily on customer survey data or focus group feedback as a means of integrating customer preference information into the product design process. The reliance of direct customer interaction can be costly and time consuming and may therefore limit the overall size and complexity of the customer preference data. Furthermore, since survey data typically represents stated customer preferences (customer responses for hypothetical product designs, rather than actual product purchasing decisions made), design engineers may not know the true customer preferences for specific product attributes, a challenge that could ultimately result in misguided product designs. By analyzing large scale time series consumer data, new products can be designed that anticipate emerging product preference trends in the market space. The proposed data trend mining algorithm will enable design engineers to determine how to characterize attributes based on their relevance to the overall product design. A cell phone case study is used to demonstrate product design problems involving new product concept generation and an aerodynamic particle separator case study is presented for product design problems requiring attribute relevance characterization and product family clustering. Finally, it is shown that the proposed trend mining methodology can be expanded beyond product design problems to include systems of systems design problems such as military systems simulations.

To my parents and siblings, for their unconditional love and support

Acknowledgments

I would like to thank my advisor, Professor Harrison Kim for his guidance and support throughout the years. His dedication and vision served as a true inspiration for this work and I am grateful to have had him as an advisor. I extend my sincere appreciation to my committee members, Loretta Auvil (NCSA), Professor Deborah Thurston (IESE), and Professor Yuanhui Zhang (ABE) for their dedication and fruitful discussions that have enhanced the quality of this work. I would like to thank the Gates Millennium Scholars (GMS) foundation for their unprecedented financial support and motivation throughout the years. I would like to thank Sandia National Labs for their generous support from 2008-2010 and for providing the necessary simulation and analysis tools for certain components of this research. Undergraduate researchers also played a vital role in the software development of some aspects of this work. I would like to acknowledge Yohannes Kifle and Sukolsak Sakshuwong for the programming aspects and software development of this work that helped validate some of the proposed methodologies. I would also like to thank the Department of Industrial and Enterprise Systems Engineering (IESE) for granting me the opportunity to grow as a researcher and scholar.

This work has also been supported by the National Science Foundation under Award No. 0726934. I would like to thank the NSF CIEG program for providing the High Performance Computational resources to conduct this research. A special thank you to Dr. Natasha Balac for being such a great mentor during the CIEG summer project. In addition, I would like to thank to the SDSC CIEG program director, Dr. Amitava Majumdar, and researchers Dr. Jerry Greenberg, Dr. Hector Jasso, Dr. Mahidhar Tatineni and all of the other researchers and staff that added to the quality of this research.

Finally, I would like to thank Stephanie Thea for her patience and support while I pursued this degree and for being there during my trials and tribulations.

Any opinions, findings, and conclusions or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the National Science Foundation, Sandia National Laboratories, Gates Millennium Scholars (GMS) foundation or other relevant supporting entities.

Table of Contents

List of Tables

List of Figures

List of Symbols

- γ: The maximum predicted price (MaxPrice) a customer is willing to pay
- Λ_i: Attribute selection which can assume a range of values a_i
- K: Product portfolio limit (Maximum number of existing products at launch)
- π: Profit of product variant based on engineering design and predicted demand
- T^C: Architecture target component predicted by data mining product preference model.
- R^E: Engineering response component cascaded up to the enterprise level
- y: Linking variable at the engineering sub-system level cascaded up to enterprise level
- ε_R: Deviation tolerance between customer performance targets and engineering response
- ε_y: Deviation tolerance between linking variables

Chapter 1

Introduction

The emergence of highly competitive markets in the global market space have caused companies to reevaluate strategies for ensuring sustainable business endeavors. Attempts to satisfy a wide array of customers quickly and efficiently have lead to the concept of product customization, wherein enterprise decision makers strive to better cater to the needs of their customers through a wider array of products to choose from [3]. In this product design and development paradigm, the physical and functional characteristics of products are more flexible to address the rapid changing customer needs [4]. Though the strategy of mass customization has great potential to increase market share and enterprise competitive advantage, efficient execution and management of such large scale operations can prove quite cumbersome, especially during the manufacturing process reconfiguration. In an attempt to mitigate some of the costs associated with customization, companies have focused on commonality among product variants [5].

Commonality among product variants has the potential to reduce the design and manufacturing complexities that may arise due to mass customization. Products designed around a shared and efficient *product architecture* can reap the benefits of *economies of scale* [6] that exist due to a consistent and stable manufacturing process. The term *product architecture* is frequently defined as the set of modules/components wherein product variants evolve [7, 8, 9]. In many instances, product variety and product commonality are competing objectives. On one hand, the more diverse a product portfolio is, the fewer shared components within the product family. On the other hand, the more similar a product portfolio is, the greater the potential for shared components within the product family. These tradeoff decisions should be justified based on the level of customer satisfaction and overall enterprise profits. There have been many proposals for determining customer preference information and the degree of commonality in product design and development. However, the lack of standard performance metrics to evaluate product family decisions has hindered consensus in this field [10].Companies continue to place a high premium on the methodologies needed to ensure that mass customization decisions lead to increased, consistent profit margins.

1.1 Objective and Scope

This research aims to develop a Multidisciplinary Design Optimization (MDO) approach to product design through a synergistic methodology that incorporates the objectives of each discipline (marketing, engineering design, manufacturing, distribution, etc.) into the realization of an optimal product portfolio. As consumers' demand for customizable products continue to increase, companies are faced with the complex challenge of balancing design and manufacturing costs. The product family paradigm allows companies to design products around a shared and efficient product architecture. This allows for cost savings benefits to be realized through economies of scale that exist as a result of component sharing decisions. Commonality decisions between products can however lead to decreased product performance and ultimately, diminished customer demand. With the incorporation of predictive data mining techniques, this research merges customer preferences directly with engineering design to achieve an optimal family of products. This process is an iterative approach that employs the decomposition and integration techniques of multilevel optimization.

Although this research has focused primarily on machine learning in the context of product design, it can be extended to other complex systems design problems involving large scale data generation. One such expansion has been the application of the aforementioned machine learning approaches to Systems of Systems research involving the simulation of military operations. Through collaborative research with Sandia National Laboratories, it has been demonstrated that machine learning techniques can be employed in Systems of Systems problems to help quantify critical military systems relevant to the overall mission success. The newly proposed Trend Mining algorithm that has been employed in product design scenarios will be extended to include time varying military simulations to help discover emerging systems trends.

1.2 Motivation

The challenges facing enterprise decision makers in the product portfolio development process are multifaceted, including identifying candidate product concepts that have the greatest probability of market success. Attempts to design and produce every possible product concept may be impractical in real life design processes, especially when *first to market* may create tremendous competitive advantages in the market space. In such highly dynamic markets, the acquisition and translation of customer wants into engineering design targets needs to be swift and efficient. Traditional customer preference techniques that have been employed in the design community have relied on customer survey approaches to quantify customer wants. The time and resources needed under these traditional customer preference techniques may limit the size and complexity of the demand modeling process and hence, adversely affect the engineering design of next generation products. Moreover, traditional demand modeling techniques frequently em-

ployed in the product design community typically generate predictive models using data from a single snapshot in time (usually the most currently available data set) and hence may not reflect the evolving nature of product trends. The absence of a temporal demand model for product design presents a challenge to design engineers trying to determine the relevant product attributes to include/exclude in the next generation of products.

To overcome the aforementioned limitations of current product design methodologies, this research integrates large scale, time series data sets of customer preference and extracts meaningful product attribute information to help guide the product design and development process. The overall objective of maximizing company profit is realized when a feasible set of product variants is presented in the final solution process.

1.3 Literature Review and Related Work

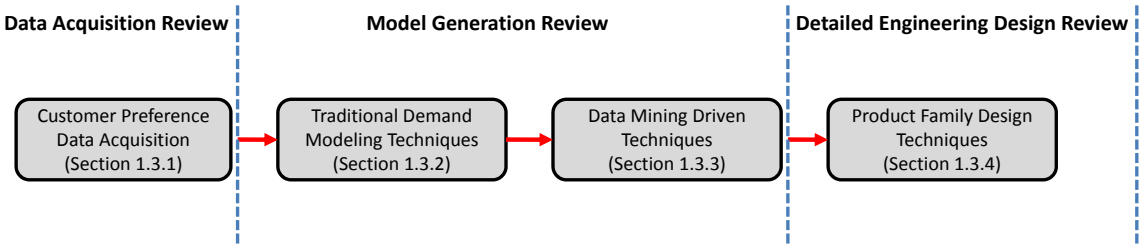


Figure 1.1: Sequence of topics discussed in the Literature Review and Related Work section

The literature review in this dissertation is partitioned into 3 sections as illustrated in Figure 1.1: Data Acquisition Review, Model Generation Review, and Detailed Engineering Design Review. Each section discusses the state of the art and formally introduces concepts and terminology that will be relevant in understanding the proposed data mining driven product design methodologies introduced later in this work.

1.3.1 Customer Preference Data Acquisition

Preference data frequently used in demand modeling techniques can be characterized as either *revealed data* or *stated data*. *Revealed*preference data can be thought of as data that is acquired directly from actual customer actions and can be used to estimate future customer behavioral patterns [36]. Revealed preference data can be acquired through credit card transactions, online purchases, and in-store transactions. *Stated* preference on the other hand deals more with hypothetical purchasing scenarios and is more survey focused [37]. Customer data is acquired through direct interaction where customers state their preferences based on the perceived value of a product. In *stated* preference data, attributes may be more theoretical as this type of data is typically used to test novel attributes and new product concepts. The need for close customer interaction with the stated preference data therefore limits the size of the data

set compared to revealed data that can comprise of extremely large data sets of electronic customer data.

1.3.2 Traditional Demand Modeling Techniques in Product Design

Quality Function Deployment

The Quality Function Deployment (QFD) is a design and development methodology that attempts to acquire customer requirements (CR) otherwise known as the Voice of the Customer (VOC) and translates them into functional engineering targets [11]. A conventional approach to customer preference acquisition is through focus group interviews or surveys of a sample of current or future customers [12]. Corresponding weights are assigned to each customer requirement based on an importance rating indicated by a customer [13, 14]. A QFD matrix is often used to depict the interdependence between customer requirements and the engineering metrics (EM) and aid in brainstorming and designing the optimal product to address customer needs. QFD driven product development methodologies suggest that QFD is well suited for out of the box solutions to customer needs due to the fact that engineering design features are evaluated based on their positive and negative contributions to solving the product design problem. The design of the QFD matrix also makes it easier to benchmark a particular design solution against competing brands. Since the QFD methodology is based on direct customer feedback data, the size of the data set as well as the number of attributes studied may not capture the overall requirements of the market space. Furthermore, the customer importance weighting of attributes is somewhat qualitative in nature and nonuniform amongst respondents. These limitations of QFD based methodologies in product design serve as motivation for the data mining based product design methodology that is proposed in this work.

Conjoint Analysis

Conjoint Analysis (CA) has been used successfully in marketing to determine how customers value combinations of different product attributes/features [11]. In this approach, a target customer group is identified for the study and presented with a set of attributes (survey format or prop cards [15]), each with different levels (attribute ranges) [16]. *Part-worth utilities* are estimated based on customer importance ranking of individual product attributes. The resulting utility function is used to evaluate customer preferences for different attribute combinations. Although Conjoint Analysis application areas can range from human psychology to advertising, attempts to directly incorporate it into engineering design optimization and product development have been investigated [17, 18, 19, 20]. These Conjoint Analysis based product development methodologies highlight the ability of the approach to quantify specific product attribute levels in new product development. This approach is primarily survey driven and therefore as the attribute space becomes large, preserving the quality of the model becomes a challenge.

Discrete Choice Analysis

The Discrete Choice Analysis (DCA) approach has been employed extensively in the product design community to model consumer choice behavior from a set of mutually exclusive, collectively exhaustive choice alternatives [21, 22, 23, 24]. DCA incorporates probabilistic choice theory in determining which product a customer is most likely to choose, based on expected utility [25]. DCA employs a random utility to measure the variations in customer preferences. Given m choice alternatives, it is assumed that a customer n enjoys a certain utility U_{ni} of a given choice alternative i, i=1, .., m [26]. From a research standpoint, only the observed utility for a customer can be estimated, resulting in error due to utility not observed by the researcher. The utility function U_{ni} therefore contains a deterministic observable part (W_{ni}), and an unobservable random part (ε_{ni}) which constitutes the error. The three fundamental assumptions of the DCA are the following: (1) The choice set contains finite, quantifiable alternatives. (2) The alternatives within the *choice set* are mutually exclusive, that is, choosing one alternative implies that no other alternative within the choice set is selected. (3) The choice set is collectively exhaustive meaning that all possible alternatives are included within the choice set. While there are a number of DCA techniques popular in literature (e.g., Multinomial Logit, Nested Logit, Mixed Logit), they are distinguished from each other by the degree of sophistication with which they model the unobserved error and heterogeneity in customer preferences. In this work, the Multinomial Logit (MNL) choice model is employed. The MNL choice model is also known as the conditional logit model, which differs from standard Multinomial Logistic Regression (abbreviated MNR in this work). The MNL choice model assumes that a respondent's choice is a function of both the attributes of the product and the customer, rather than the attributes of the customer only as assumed in the MNR model. The MNL model assumes that the error terms (ε_{ni}) found in the utility function in equation (1.1) are independent and identically distributed (i.i.d) and follow a *Gumbel* distribution. This is represented by:

$$U_{ni} = W_{ni} + \varepsilon_{ni} \tag{1.1}$$

The deterministic part (W_{ni}) of the utility function (U_{ni}) can be represented by the observable independent variables $\mathbf{Z}$ and the unknown coefficients β as given by the equation below [26]:

$$W_{ni} = f(A_i, P_i, S_n : \beta_n) \tag{1.2}$$

Where,

$\mathbf{A}_i$: Represents the quantifiable product attributes desired by the customer

P_i: Represents the price for a given product (choice alternative i)

$\mathbf{S}_n$: The socio-demographic attributes of a customer

β_n: The unknown coefficients representing a customer's taste preference

Estimation of the observed utility function W_{ni} allows for the choice probability of a design alternative to be estimated. The probability that a customer n will choose alternative i over alternative j ($i \neq j$) from a choice set C_m is mathematically defined as [8, 20]:

$$P_n(i = C_m) = \frac{e^{W_{ni}/u}}{\sum_{j=1}^{m} e^{W_{nj}/u}} = \frac{e^{\beta_n^T z_{ni}/u}}{\sum_{j=1}^{m} e^{\beta_n^T z_{nj}/u}} \tag{1.3}$$

Where,

P_n(i=C_m): Represents the probability that a customer n chooses alternative i within the choice set C_m .

β_n: Unknown parameters estimated using the Maximum Likelihood Estimation (MLE)

$\mathbf{Z}_{ni}$: Represents the observable independent variables which include the product attributes (**A**), the customer demographic attributes (**S**) and the price of the product (P).

u: scaling parameter set to 1, assuming all choice alternatives are equally considered by customer n.

1.3.3 Data Mining in Product Design

Static Machine Learning Models

The acquisition of customer preference data to determine patterns is vital to the overall stability and success of a company's next generation product portfolio. Stored data can be related to manufacturing capabilities, consumer tendencies, distribution patterns, sales, etc. [27]. To this end, automated analysis and discovery tools that are powerful enough to analyze large data sets are becoming more popular. As data storage and information retrieval capabilities become more widely available, there is an emerging trend for companies to acquire and store customer preference data. For example, the physical characteristics (vehicle horsepower, number of doors, color, etc.) of an automobile purchased by a customer visiting a dealership, along with the customer's demographic information (age, gender, household income, etc.) can used to determine emerging trends in the automotive industry and design next generation products accordingly. A great challenge in storing such data for product design purposes is the non-homogeneity of customers preferences. Consequently, as the size of non-homogeneous data increases, so does the complexity of identifying natural patterns within the data set. The ability to determine suitable product architectures for a particular group of customers becomes a challenge as enterprise decision makers and engineers attempt to extract meaningful patterns within the data set to aid in the product design and development process. Data mining in the context of product development is an emerging area of research that has the potential to significantly impact engineering design and manufacturing efforts [28, 29]. By identifying patterns within the large data set of customer preferences, engineers can incorporate this knowledge in the product family design process.

The incorporation of data mining techniques in product portfolio development is emerging as a well-founded approach to extracting and analyzing relevant customer information. Kusiak and Smith highlight several key areas in industrial and manufacturing design processes where data mining techniques could potentially have great benefits [30]. In the context of product portfolio development, the application of data mining clustering techniques in the design of modular products has also been investigated. Moon *et al.* use data mining to represent the functional requirements of customers and use fuzzy clustering techniques to determine the module composition of a product architecture [3]. Nanda *et al.* propose a Product Family Ontology Development Methodology (PFODM) that utilizes a formal concept analysis approach in the design of product families [31]. This approach incorporates existing knowledge of the product family in generating a hierarchical conceptual clustering of design components. Agard and Kusiak [30] employ data mining clustering techniques to segment a customer data set into candidate target markets for the design of product families. Association rule mining is then used to determine attribute patterns in the segmented data. Menon *et al* propose applying a textual based data mining approach to search large data sets within different stages of the product development process in order to reduce human error in the analysis process [32]. The association rule mining algorithm with *support* and *confidence* levels is used to identify patterns that exists in a call center database case study. Su *et al* approach the customer segmentation problem in product development through a combination of data mining clustering techniques; *k-means*, self organizing map (SOM) networks and Fuzzy Adaptive Resonance (FuzzyART) theory [33].

Time Series Machine Learning Models

The area of data mining dealing with dynamic information processing is relatively new and has great potential to address many challenging areas of research. *Change Mining* is the umbrella term used to describe research involving data evolution in dynamic data bases [34]. *Data Stream Mining* is a subcategory of Change Mining that deals more with the continuous flow of data that needs to be analyzed with limited memory complications.

There have been several data mining algorithms proposed to address continuously-changing data streams. For example, the Very Fast Decision Tree (VFDT) learner employs the Hoeffding statistic to build a decision tree classifier that has similar predictive characteristics as a conventional decision tree learner (the C4.5 or gini based decision tree learners) but with a fraction of the memory requirements [35]. Another example is the Concept-adapting Very Fast Decision Tree (CVFDT) which extends the capabilities of the VFDT by enabling it to accommodate time-sensitive streaming data that may tend to exhibit *concept drift*. Concept drift is a phenomenon in dynamic information processing where the target variable shifts over time and causes the data mining model to diminish in its predictive accuracy [36]. While these models have the ability to handle incoming data streams, they are more focused on generating/adapting a model based on incoming data, rather than understanding how the data patterns evolve altogether.

Research domains more interested in data trends, rather than the speed of the data streams also present another

interesting area of study. For example the *RePro* classifier is a data streaming algorithm that applies both proactive and reactive predictions during model generation [37]. The algorithm attempts to alleviate the problems of *concept drift* by anticipating concept changes and making predictions that if incorrect, cause the model to readjust and revert back to a previous model. Another example is the *PreDet* algorithm that fits a polynomial regression model to the monotonically increasing or decreasing time series attribute relevance statistics. The resulting time series model anticipates future attribute patterns that are inherent in the evolving data [34].

In the product design domain, there has been little research in design methodologies that have the ability to capture the evolving nature of customer preferences. The Preference Trend Mining (PTM) algorithm that is proposed in this work differs from the *PreDet* and other change mining algorithms by having the ability to anticipate emerging attribute behavior whether the attribute exhibits a monotonically increasing or decreasing trend, cyclical trend, or no trend at all. In addition to this, the aforementioned change mining algorithms do not suggest approaches to characterize attributes that may exhibit weaker predictive power over time. The notion of attribute *irrelevance* is handled by classifying attributes based on their time series predictive power. This enables the PTM model to accommodate attributes that may be experiencing changes in the distribution of the attribute values themselves or novel/emerging attributes. The goal of the proposed PTM algorithm is to enable design engineers to understand changing customer preferences and anticipate emerging product design trends in a timely and efficient manner.

1.3.4 Product Family Design

Once customer preference data has been acquired and a predictive preference model has been generated for new product designs, engineers are then faced with the challenge of designing product portfolios that meet customer preferences in a cost effective and efficient manner. The product family paradigm has been proposed to address the challenges of designing products for mass customization or for highly diversified customer functionality requirements. The term *product family* is frequently defined in literature as a group of related products that share an underlying product design architecture [38, 39, 29]. The product family paradigm enables companies to standardize certain aspects of a product and at the same time provide product diversity to customers through product variants. Product cost savings may be realized as a result of product standardization due to *economies of scale* (e.g., cost savings due to a standard manufacturing line for all products, rather than a specialized manufacturing line for each product). However, greater product standardization may also lead to lower product diversity in the market space and diminished product performance (e.g., limited customizable features for customers such as product color, reliability, size, etc.) Therefore, in the product family approach, the level of product standardization versus product variety presents a trade-off scenario as product performance and appeal (from the customer's perspective) may diminish in an attempt to increase product standardization [38].

The product family design problem has been segmented into two well established domains: the *Bottom-Up* approach and the *Top-Down* approach [39]. In the Bottom-Up approach, companies are more interested in making significant improvements to an existing product portfolio by combining products within the existing product family into a new product architecture. An assumption in the *Bottom-Up* approach is that the newly redesigned product family will be able to satisfy customer needs through minimal additional technology investments. On the other hand, in the *Top-Down* approach to product family design, the next generation of products is not based on an existing product family, but instead emerges from a market driven need. This need arises from the evolution of customer preferences far beyond what the current product portfolio can satisfy [39]. In this work, a *Top-Down* product family methodology is proposed that analyzes large customer preference data sets and identifies candidate product architectures that will be used in the product family design. This product design architecture can represent a group of design components that perform a series of functional processes. Products sharing similar product architecture can satisfy a broader range of customer requirements simply by possessing functionality capabilities that vary beyond the underlying architecture. The sharing of components also has the potential to reduce the time and costs associated with manufacturing diverse products. The challenges that face such decisions include designing product variants with an excess of common components that ultimately decrease the individual performance aspects required by customers. Therefore tradeoffs have to be made in regards to the level of commonality exhibited by product families [40]. A major externality that drives commonality decisions is the added pressure by customers that have established product customization as a standard market requirement.

de Weck and Chang propose an approach that utilizes sales volume sensitivities and product variant performance to dictate the number of optimal product architectures existing in the product portfolio [41]. Messac *et al.* employ a physical programming based approach that focuses on scale-based product family optimization. In this approach, the concept of *scaling* is used to change product specifications due to external factors [38]. Gonzales-Zugasti *et al.* use an *Interactive Implementation* approach that first establishes a product architecture design, then its variants [42]. The assumption of a pre-existing product architecture significantly constrains the engineering design team in attempting to satisfy external performance requirements. Other approaches by Desai *et al.* [43] and Kim and Chhajed [44] partition the consumer market into two groups: high end and low end customers, and design product variants based on the performance and quality expectations desired by each market. These approaches to some extent incorporate customer input in the design of product families but greatly restrict the customer pool by only partitioning the market into high-end and low-end segments, hereby possibly overlooking customers in between these two segments.

Commonality in Product Family Design

Commonality among product variants is a widely acceptable method of mitigating the inevitable cost increases of such highly differentiated products. By designing product variants around a shared and efficient product architecture, companies can reduce design and manufacturing costs associated with product differentiation [5, 45]. Commonality indices in product family formulation can be traced back to works by Collier in his proposal of the *Degree of Commonality Index* (DCI) [46, 47]. The DCI is the ratio between the number of common components in a product family and the total number of components. Mathematically, this can be represented as:

$$DCI = \frac{\sum_{j=i+1}^{i+d} \phi_j}{d} \tag{1.4}$$

Where

- ϕ_j: Represents the number of immediate parents that component *j* has over a set of end items or product structure level(s).
- d: Represents the total number of distinct components in a complete product
- i: Represents the total number of end/highest level parent items for the product structure level(s).

The DCI can vary between *1* and β, where

$$\beta = \sum_{j=i+1}^{i+d} \phi_j \tag{1.5}$$

That is, if DCI=1, component sharing is nonexistent in the product family and products comprise of individually unique components. On the other extreme where DCI=β, then all of the components are shared within the product family [46]. The relative simplicity of the DCI formulation makes this a less computationally complex index for evaluating product commonality decisions and therefore may be quite beneficial for low fidelity optimization models. One major limitation of this approach however is the difficulties in evaluating commonality decisions across highly diverse product families. Later works by Collier propose an improvement to the DCI metric by introducing the *Total Constant Commonality Index* (TCCI) [37]. In this formulation, an absolute bound is set within [0,1] to gain

a quantifiable measurement of commonality decisions. In this updated index, increases in the degree of commonality are easier to measure. Other contributions such as that proposed by Jiao and Tseng [48] expand on the DCI by Collier to include production, operating, and component costs.

With the *Product Line Commonality Index* (PCI) introduced by Kota, *et al* [49], product families comprising of highly differentiated products are penalized on the basis of lack of component similarity. This is a departure from the DCI that penalizes for product variety, which in some cases may be what the customer market demands.

In the product commonality approach adopted by Fellini *et al.*, null platform optimal designs are first identified and with the incorporation of sensitivity information of the design variables [50, 8]. A performance deviation vector (Π) is then computed, with component sharing decisions subsequently made on the basis of the acceptable performance deviation from the performance deviation vector [8].

The *Percent Commonality Index* (%C) partitions the commonality problem into 3 separate categories: Namely, *Component*, *Connection* and *Assembly* [51]. The number of similar components present in a product family is represented by the *component* commonality while the *connection* commonality represents the level common connections between components. The *assembly* commonality index is similar to the previous, as it measures the level of common assembly workstations [47]. The combination of these three metrics gives the commonality evaluation for a particular architecture. Similar works by Martin and Ishii [52] define a *Commonality Index* that identifies unique components as the basis of commonality decisions.

More recent contributions to the product family commonality paradigm are introduced by Thevenot and Simpson [53] with their proposed *Comprehensive Metric for Commonality* (CMC). The CMC approach classifies components based on its cost C_i and f_{ji} factors, as defined by the methodology [53]. A predefined rating criteria is used to reward/penalize commonality decisions based on the CMC.

1.4 Overview of Data Mining Driven Design Methodology

The proposed product design research methodology aims to systematically link customer product preferences with the enterprise decision making processes in an attempt to design an optimal portfolio of products that meets enterprise level targets, while at the same time satisfying customer needs. The customer preference modeling is based on data mining/machine learning principles while the engineering product validation is formulated as a multi-level optimization model. This multi-level optimization strategy aims to maintain consistent interaction among all facets of the product development process (customer predicted targets, product component sharing, enterprise profit objectives, etc.). The process from data extraction to predictive product design model is illustrated in Figure 1.2, with the detailed description of each step presented below.

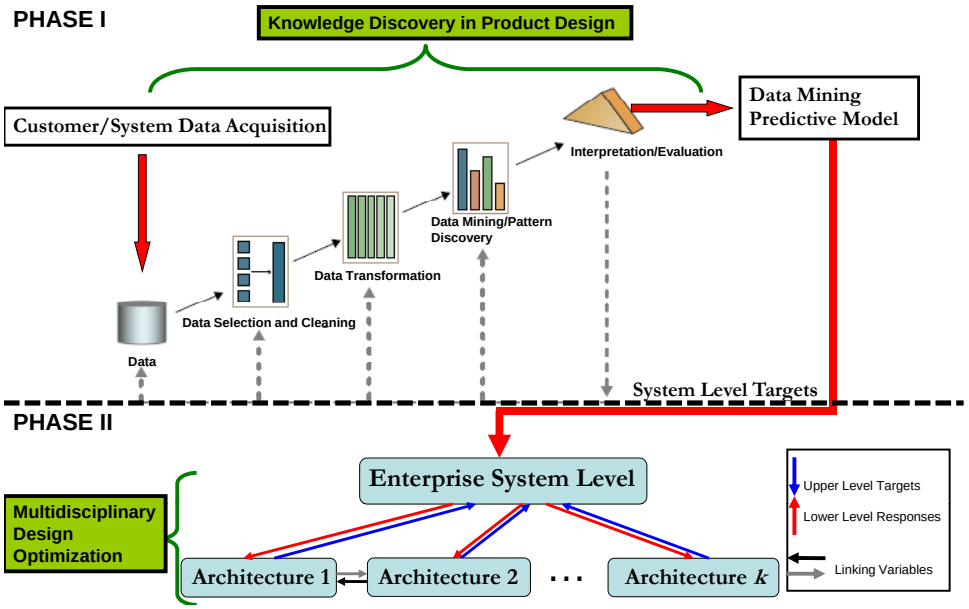


Figure 1.2: Overall Predictive Product Portfolio Formulation (Adapted from D2K manual [1])

1.4.1 PHASE 1: Customer Knowledge Discovery

Knowledge Discovery in Databases (KDD) has become known as the nontrivial means of extracting information in large scale databases that were previously too complex for human analysis [54]. As data extraction and storage capabilities become cheaper and more readily available, there is tremendous opportunity for design engineers and enterprise decision makers to incorporate this vast wealth of knowledge directly into new product development efforts. Data mining is a step along the KDD process that utilizes machine learning algorithms to extract meaningful, previously unknown information from large data sets [27]. The concept of data mining can be applied to product portfolio formulation wherein the exact product specifications and manufacturing quantity (predicted demand information for each individual product concept) can be determined directly from data mining predictions. The process from data extraction to predictive model is as follows (Figure 1.2):

Step 1: Data Acquisition

The acquisition and storage of data is paramount in the product portfolio formulation process. The process begins by acquiring the raw data set to be used in the data mining sequence. This data can be acquired in several ways. One approach is by conducting a realistic customer survey to capture customer product preferences, then translating these preferences into meaningful engineering design targets [55]. Another approach would be for this data to already exist in a data warehouse, i.e., stored data from past customer purchasing behavior (e.g., *SQL Server*) [56].

Traditional Database Management Systems (DBMS) can be found in many of today's large corporations and have

become the foundation of modern day data storage and analysis [57]. The success of the internet has increased the importance of Database Management Systems in areas such as business transactions, communication networks, social networks, to name but a few. DBMS have evolved since the early concepts proposed in the early 1960s that were based on the CODASYL query language [58]. A significant advancement in DBMS came when Ted Codd proposed a novel representation of data as relations that could then be queried using a high level language such as the Structured Query Language (SQL) [59]. The Entity-Relationship (ER) [60] model proposed by Peter Chen in subsequent years helped compliment the original Relational model by providing a providing a more microscopic representation of the interactions within the data itself. Together, both the Relations model and the ER model form the foundation of many of today's large scale Database Management Systems.

Step 2: Data Preprocessing (Data Selection and Cleaning)

Once the data has been captured from Step 1, Step 2 is where irrelevant or noisy data is identified and removed and relevant data is extracted from the raw data [56]. There are many well established approaches that deal with missing attributes or ambiguous responses ranging from *most common attribute, event covering method, or ignoring the value altogether* [61, 62, 63]. When dealing with electronic transactional data (online, in-store), it is then possible to collect, clean and store this data in a *data warehouse*. A data warehouse is a preprocessing stage that integrates all data into one source (this includes, raw data, historical data, summarized data, etc.)[64]. The accuracy of the data mining model will be highly dependent on the data selection and cleaning step and it is therefore important that considerable time is allocated to preparing high quality data for the pattern discovery step that follows. There are many algorithms that exist in today's data mining analysis tools that are now capable of incorporating this data selection and cleaning process directly with the overall knowledge discovery process [1, 65].

Step 3: Data Transformation

The final data preprocessing step involves transforming the data into acceptable forms for the appropriate mining algorithm. Data transformations can include *binning, normalizing, missing value imputation, etc.* [66]. This can either be done manually by the user or automatically by a data mining analysis tool [1].

Step 4: Data Mining/ Pattern Discovery

The majority of the work in this dissertation focuses on proposing specific Data Mining/ Pattern Discovery methodologies to address a variation of product design challenges. Engineers will be able to assess their existing engineering design needs and employ one of the proposed product design methodologies below to help solve them:

- *Naïve Bayes Classification for Flexible Product Design*: For design problems involving flexibility in the product attribute space, the Naïve Bayes machine learning approach is employed to help design engineers understand the implications of attribute inclusion/exclusion in new product design. That is, given a large customer data set containing revealed preference data (customer product preferences acquired after a customer has made a purchasing decision, ex: credit card transactions, store purchases or online transactions, etc.), design engineers can include/exclude certain product attributes and observe the subsequent consequence on the overall product design objective. The final set of attributes is then linked with a multilevel optimization model for product design optimization.

- *ReliefF and X-means Clustering for Product Family Design*: For product family design problems involving multiple products, design engineers may not always know the total number of products that would satisfy a specific subset of customers. Furthermore, design engineers may want to quantify and rank the relevance of attributes so that product designs can be more customizable. The ReliefF attribute weighting machine learning technique is proposed to help design engineers quantify attribute relevance to the overall product design decisions made by customers. The X-Means clustering technique can then be employed to group similar product designs based on the overall structure of the given product data, rather than an approximation by engineers. As a result, the appropriate number of products within a product family can be identified and designed around a shared and efficient product architecture.

- *Decision Tree Classification for Product Concept Generation*: The C4.5 Decision Tree Classification technique has been employed for solving product concept generation problems where design engineers are primarily focused on identifying *relevant* attribute combinations pertaining to a given design objective. This iterative machine learning technique generates a set of decision rules that can be then used to guide new product designs, modeled as a multilevel optimization problem. This approach is well suited for large scale customer data involving a high dimensional attribute space as the resulting decision tree will only include attribute combinations that are strong indicators of an overall product design objective.

- *Trend Mining for Predictive Product Design*: Although several machine learning techniques exist dealing with temporal data, there exists the need for a data mining algorithm that specifically captures evolving customer preference in product design. To solve this, a trend mining algorithm is proposed to help quantify and model changes in customer preference towards given product design attributes. An attribute classification model is proposed to help design engineers make decisions involving attributes that are deemed irrelevant by a machine learning algorithm. That is, if an attribute is consistently characterized as irrelevant to the overall product design objective, design engineers can consider this attribute *obsolete* and exclude from all future product designs,

standard and include in future product designs (regardless of its irrelevant characterization by the machine learning model), or *non- standard* and realize that such an attribute has not been fully adopted/understood in the market segment and therefore has no consistent effect on product design decisions.

- *Capturing Product Preference Trends Using Publicly Available Customer Review Data*: A major challenge that has plagued the product design community has been the lack of large scale, realistic customer data to validate proposed product design methodologies or solve complex design problems. Researchers and design engineers often resort to conducting customer surveys or focus group interviews that can be costly and time consuming. Recent research studies have shown that a major online retail company can have as many as 10 million active product reviews while it has been reported that more than 50% of online customers indicated that customer reviews played an important role in influencing their purchasing decisions. Based on these findings, a publicly available online interface (www.trendminingdesign.com) has been developed that is built upon an automated computer exploration algorithm that captures and stores time series product preference data. This research aims to provide design engineers with direct access to large scale, time series customer preference data that can be used to guide next generation product designs.

Step 5: Interpretation and Evaluation:

The results from the Knowledge Discovery phase will aid in the next generation product design process by systematically linking novel, previously unknown customer preferences with a detailed engineering design model for product design validation (Phase 2).

1.4.2 PHASE 2: Linking Customer Knowledge Discovery with Product Design

Solving Product Family Design Problems Through Multidisciplinary Design Optimization Techniques

With the incorporation of predictive data mining and machine learning techniques, this research merges customer preferences directly with engineering design to achieve the most profitable, and concurrently the most desirable family of products. This process is an iterative approach that employs the decomposition and integration techniques proposed to solve Multidisciplinary Design Optimization problems. Multidisciplinary Design Optimization (MDO) is a systems engineering approach to designing complex systems (such as automotive, aeroplanes, consumer electronics, etc.) across multiple disciplines in a decomposed setting [67]. This departs from conventional All In One (AIO) formulation that integrates subsystem design variables and constraints in an overall system objective [68]. The major driving force for the paradigm shift is the computational complexities that are involved in solving AIO problems involving multiple disciplines. The MDO solution approach addresses these challenges in many ways, one of which is by decomposing

the overall system objective into subsystem models that can be solved more efficiently [67].

There have been several proposed approaches to solving MDO problems such as Collaborative Optimization [69], Bi-Level Integrated System Synthesis (BLISS) [70], the constraint margin approach based on the decomposition of quasi separable problems [71], Analytical Target Cascading (ATC)[72, 73, 74, 75], Penalty Decomposition (PD) methods [76], an augmented Lagrangian decomposition method (for quasi-separable problems) [71], to name but a few. The solution approach is highly dependent on the type of problem being solved as each algorithm has its strengths and limitations.

The ATC decomposition approach has been employed to solve product family design problems, where customer predictive targets are to be satisfied by an engineering design response. The linking variable concept proposed by ATC will be utilized in the commonality and component sharing decisions discussed later in the work as well as the criteria for determining optimality (Matching customer targets with engineering response).

Analytical Target Cascading Analytical Target Cascading (ATC) is a multilevel decomposition approach for solving large scale engineering problems [77]. In the product development process, matching performance targets closely with final product functionality is paramount and hence the need for a robust design methodology to help accomplish this.

The ATC formulation process partitions the AIO problem into system and subsystem levels, enabling top level system design targets to be cascaded down to lower subsystem level. The steps are [75]:

1. Development of appropriate analysis models
2. Partitioning the system
3. Formulating the target problems for each element of the partition
4. Solving the partitioned problem through a coordination strategy to compute all stated targets

ATC has also been extended to product families through the linking variable concept wherein subsystems attempt to share common design variables [78]. A predefined architecture is presented and products attempt to share common components while still maintaining a consistent hierarchical ATC formulation. The goal of this approach is to realize the benefits of commonality of parts among variants. This is modeled by the shared linking variable in the bi-level quasi-separable problem formulation that attempts to achieve an optimal design solution for each product while concurrently satisfying specific product functionality requirements [76]. The term *quasi-separable* is used to denote independent subproblems that *share* a common design variable/component. In this work, subproblem simply means a unique product design. The *bi-level* formulation is used in this work to coordinate these sharing decisions among subproblems. Therefore, the individual subproblem formulation for the bi-level quasi-separable problem is as follows:

Minimize $\mathbf{f}_k\ (\mathbf{y}_{s,k}\ ,\mathbf{x}_k)$

Subject to:

$$\mathbf{g}_k(\mathbf{y}_{s,k},\mathbf{x}_k) \leq 0 \tag{1.6}$$

$$\mathbf{h}_k(\mathbf{y}_{s,k},\mathbf{x}_k) = 0 \tag{1.7}$$

For the quasi-separable formulation, each $\mathbf{x}_k$ represents the vector of local design variables unique to each subproblem (k), where k=1,..., K subproblems. The vector of linking variables $\mathbf{y}_{s,k}$ makes the subproblems quasi-separable as each subproblem sharing a linking variable becomes influenced by the solution of other subproblems sharing the same linking variable. A master problem is used to coordinate the linking variable among subproblems and is explained in more detail in each chapter of this dissertation where the ATC decomposition technique is employed.

The Analytical Target Cascading formulation described above will serve as the multi-level optimization framework in this dissertation and will be used to solve the data mining driven product family/portfolio design problems.

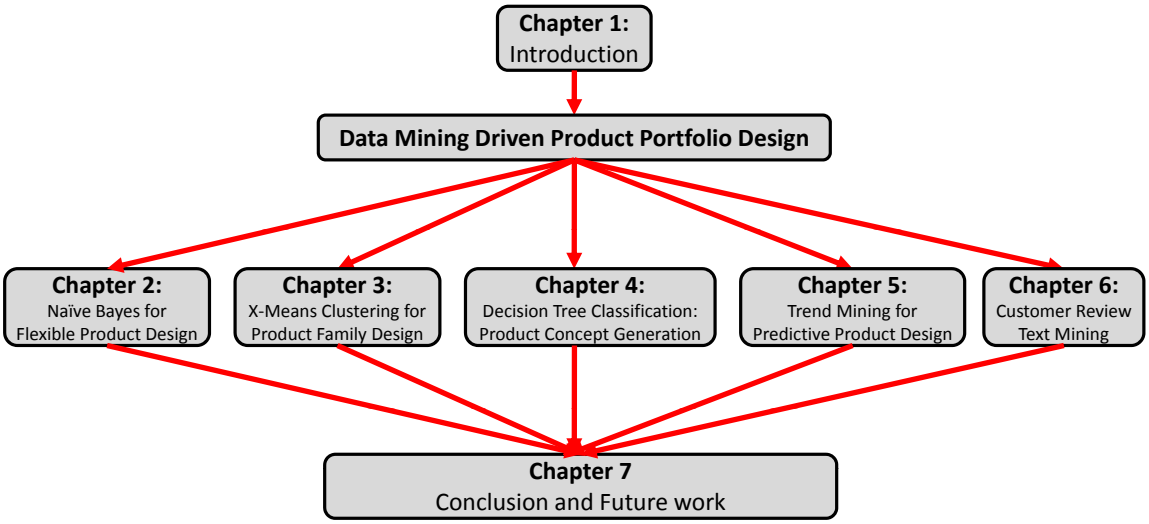


Figure 1.3: Overall organization of dissertation

1.5 Overall Organization

This dissertation is organized as seen in Figure 1.3. Chapter 1 provides an introduction and motivation of this dissertation. Chapter 2 proposes a Naïve Bayes product design methodology to address product design problems involving flexibility in attribute selection. Chapter 3 proposes a machine learning clustering technique for product family designs involving multiple operating states. Chapter 4 presents a product concept generation methodology for large scale consumer preference data. Chapter 5 proposes a data trend mining algorithm for capturing consumer preference trends and attribute relevance over time. Chapter 6 proposes a methodology of capturing product trends using publicly available customer review data. Chapter 7 concludes the dissertation and discusses future work in the research domain.

Chapter 2

Naïve Bayes Classification for Flexible Product Design

2.1 Flexible Product Design Methodology

The Naïve Bayes data mining technique is proposed for product design scenarios involving flexibility in the product attribute selection process. The Naïve Bayes data mining technique can handle large scale customer preference data, making the product design attribute selection process efficient and cost effective. In the proposed product design methodology, specific attributes can be selected manually (as the conditional probability is sequentially updated), with corresponding class prediction updated accordingly. The Naïve Bayes data mining approach to classifying is based on the fundamental assumption of attribute independence in predicting the class variable [79, 80]. The correct class prediction is achieved so long as it is *more probable* than any other class [81]. Since the Naïve Bayes class prediction is updated with each subsequent attribute selection/deselection, it is possible for enterprise decision makers to model the product design process as an optimization problem, where the overall objective would be to maximize the probability of predicting a single class variable, given a combination of attributes.

The flow diagram in Figure 2.1 illustrates the flexible product portfolio design methodology by employing the Naïve Bayes data mining technique, linked with multilevel optimization.

Naïve Bayesian Model

The Naïve Bayes algorithm builds a predictive model based on supporting evidence from a fraction (typically 2/3rds) of the customer preference data used to train the computer learning model [1]. Applied to a customer's maximum purchasing price, the Naïve Bayesian model can be posed as follows:

- Given N elements in a set of customer attributes $a_i \in A$.
- The dependent class variable **MaxPrice**(γ) has outcomes conditional on customer attributes $a_i, ..., a_N$
- $p(\gamma|a_1, ..., a_N) = p(\gamma) \cdot \frac{p(a_1,...,a_N|\gamma)}{p(a_1,...,a_N)} \longrightarrow$ Probability of MaxPrice(γ), given certain input attribute(s)
- Since the denominator of the above equation is independent of **MaxPrice** (γ) and the input attributes are known a priori, the denominator is essentially constant and can therefore be ignored [80] .

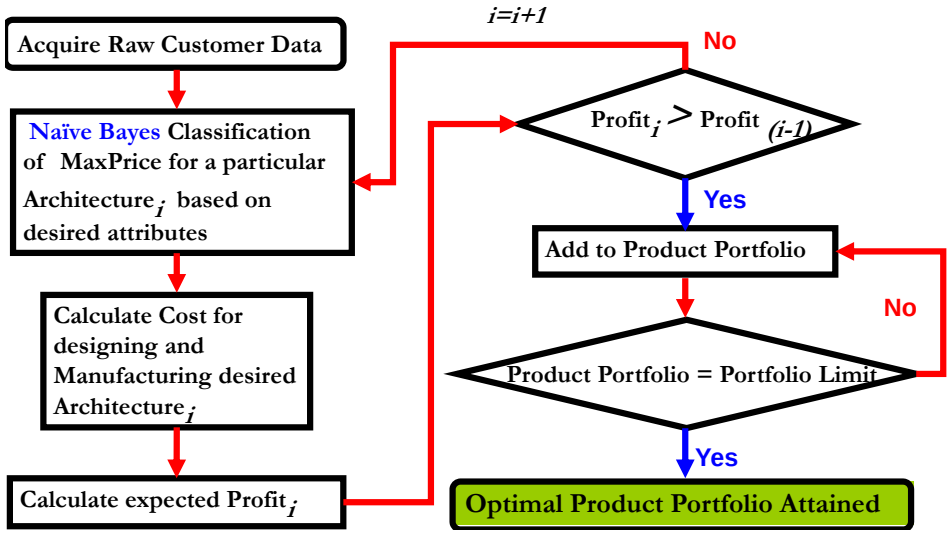


Figure 2.1: Flow Diagram of Naïve Bayes Flexible Product Design Methodology

- Using the definition of conditional probability: [82]

$$p(\gamma|a_1,...,a_N) = p(\gamma)\cdot p(a_1,...,a_N|\gamma) \tag{2.1}$$

$$= p(\gamma)\cdot p(a_1|\gamma)\cdot p(a_2,...,a_N|\gamma,a_1) \tag{2.2}$$

$$\vdots$$

$$= p(\gamma)\cdot p(a_1|\gamma)\cdot p(a_2|\gamma,a_1)...p(a_N|\gamma,a_1,a_2...,a_{N-1}) \tag{2.3}$$

The fundamental basis of the Naïve Bayesian model is the assumption of conditional independence of each input attribute, i.e. attribute a_i is independent of a_j where $i \neq j$ [80]. This is a valid assumption for the cell phone case study that will be expounded on later. For example, the assumption is made that: *The probability that a cell phone is a flip design, given a MaxPrice of $200 is independent of the probability that a cell phone has a battery life of 5 hours, given the same MaxPrice of $200.*

- The assumption of independence enables the conditional distribution of MaxPrice (γ) to be expressed as [79]

$$p(\gamma,a_1,...,a_N) = p(\gamma)\prod_{i=k}^{n} p(a_i|\gamma) \tag{2.4}$$

A machine learning approach known as *Supervised Learning* attempts to estimate the parameters of the developed Naïve Bayesian model [83]. The assumption of attribute independence allows the class variable (MaxPrice) to be estimated prior to testing the model. The Naïve Bayes Classifier combines the probability model with a decision rule;

in most cases, a most probable hypothesis rule known as *maximum a posteriori probability* (MAP) is computed, which determines the maximum likelihood of a given class [80]. The function is modeled as follows, where `arg max` is the likelihood estimator of MaxPrice [79].

$$classify(a_1,...,a_N) = \texttt{arg max}_{\gamma}\, p(\Gamma)\prod_{i=k}^{n} p\,(\Lambda_i|\Gamma) \tag{2.5}$$

- Where Γ takes on a value in the set γ, i.e. the value of MaxPrice must match a numerical value of one of the elements in the MaxPrice set.
- Similarly, Λ_i takes on a value in the set a_i, i.e. attribute value i takes on a value of an element that exists in the overall attribute set.

Prior knowledge of the attribute distribution is assumed and a point estimate of the class variable can be obtained [84]. Based on the posterior distribution, the class variable γ is estimated as the statistical mode or in other words, the most recurring. The Naïve Bayes Classifier using the *maximum a posteriori* [79] decision rule is a valid approach in the study of customer predictive preferences, as the model takes into account *a priori* [85] preference of attributes. The robustness of the classifier validates the assumption of attribute independence and correctly predicts the class variable MaxPrice [80]. The following simple example illustrates the predictive strengths of the Naïve Bayes Classifier in determining previously unknown knowledge from a given customer data set.

There are 10 unique customer responses represented by each row in the example data set (Table 2.2). The 3 attribute types (**Phone Type, Connectivity, Feature**) are mutually exclusive and comprise of binary selections. For example, the first attribute **Phone Type** can assume one of two values, (**Flip** or **Shell**), etc. The objective is to determine what combination of attributes would result in a particular class variable prediction, i.e., purchase a phone (**Yes** or **No**). For example, assume that a cell phone design having the attributes **Flip phone, Bluetooth, MP3** needs to be classified. Note that this attribute combination *does not exist* in the example data set (Table 2.2) and such a classification would therefore be considered as new, previously **unknown** knowledge [54]. To determine the class (Purchase Phone= Yes or No) that such an attribute combination would fall under, the conditional probability rule explained in equation (2.4) is employed. The conditional probabilities of each attribute are presented in Table 2.1 and the subsequent classification presented in the following calculations.

Table 2.1: Conditional probability calculations for each attribute

	Conditional Probabilities		Class Prediction
	Purchase Phone=YES (4 Occurrences)		**Purchase Phone=NO (6 Occurrences)**
P(Phone Type=Flip | Purchase Phone=YES)	**2/4**		
P(Connectivity=Bluetooth | Purchase Phone=YES)	**2/4**		
P(Feature=MP3 | Purchase Phone =YES)	**3/4**		
P(Phone Type=Flip | Purchase Phone=NO)			**4/6**
P(Connectivity=Bluetooth | Purchase Phone=NO)			**2/6**
P(Feature=MP3 | Purchase Phone =NO)			**3/6**

$$
\begin{aligned}
& p(YES|FlipPhone, Bluetooth, MP3) \\
&= p(YES) \cdot p(FlipPhone|YES) \cdot p(Bluetooth|YES) \cdot p(MP3|YES) \\
&= \frac{4}{10} \cdot \frac{2}{4} \cdot \frac{2}{4} \cdot \frac{3}{4} = 0.075
\end{aligned} \tag{2.6}
$$

$$
\begin{aligned}
& p(NO|FlipPhone, Bluetooth, MP3) \\
&= p(NO) \cdot p(FlipPhone|NO) \cdot p(Bluetooth|NO) \cdot p(MP3|NO) \\
&= \frac{6}{10} \cdot \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{3}{6} = 0.067
\end{aligned} \tag{2.7}
$$

The maximum likelihood function utilized by the Naïve Bayes model selects the class variable with the maximum likelihood of occurring, which in this case would be **Purchase Phone=YES** with a probability of **0.075**. In other words, this *new* combination of cell phone attributes has the potential of appealing to the consumer market and would therefore be a candidate cell phone design. Such powerful insights have the potential to significantly enhance the product family formulation process as attribute combinations can be analyzed and optimized to achieve a more efficient product development strategy.

This example is a simplified version of the actual customer preference data utilized in this work which comprises of a customer data set of 100,000 and a wider array of attributes. Despite such a large data set, the final Naïve Bayes predictive results took less than 300 seconds to generate, running on an Intel Pentium IV desktop (3.2 GHz).

Table 2.2: Sample customer response data

Attribute Selections | Class Variable

Customer	Phone Type	Connectivity	Feature	Purchase Phone?
1	Flip	Wifi	MP3	NO
2	Shell	Bluetooth	Camera	YES
3	Shell	Wifi	MP3	NO
4	Shell	Bluetooth	MP3	NO
5	Flip	Wifi	Camera	NO
6	Flip	Wifi	MP3	YES
7	Shell	Bluetooth	MP3	YES
8	Flip	Bluetooth	Camera	NO
9	Flip	Wifi	MP3	YES
10	Flip	Wifi	Camera	NO

Data Mining using Data to Knowledge (D2K)

The predictive model will enable a seamless translation of customer data into tangible design targets for the engineering design level. Selection or deselection of attributes to observe the effects on the class variable (MaxPrice) can be formulated as a *Mixed Integer Programming* Problem [86], where *the objective is to search through a combination of attributes that would yield the MaxPrice and market share percentages needed to maximize the overall profit of the company.*

The visual representation in Figure 2.2 is the D2K Graphical User Interface output that enables the user to manually select/deselect attributes that influence the MaxPrice prediction. The square box enclosing each attribute indicates which attributes are active in the predictive model. Only one parameter value per attribute can be active at once due to the Naïve Bayes assumption of attribute independence. The Naïve Bayesian prediction of MaxPrice has a percentage value associated with each MaxPrice prediction which translates into the percentage of customers with the same MaxPrice prediction. If none of the attributes are selected, then the MaxPrice prediction is calculated solely on the initial state of information, i.e. the surveyed customers and their overall preferences.

The active attributes in the prediction of MaxPrice are set as targets at the engineering design sub-level, while the MaxPrice is used to determine the enterprise profit for product variant i. In order to ensure an optimal product that satisfies customer wants, customer targets cascaded down to the engineering sub-level are weighted more than any other objective in the engineering sub-level such as cost minimization.

The iterative process of trying to match customer targets with the engineering capabilities yields an optimal product that is both profitable and desirable to customers. The overall process from D2K's Naïve Bayes prediction of MaxPrice

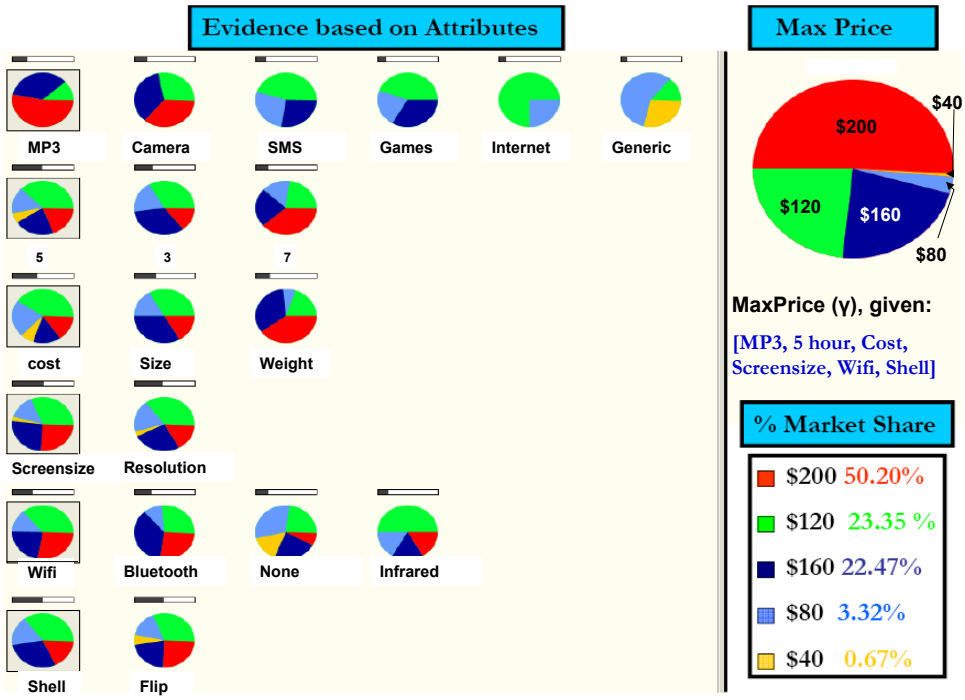


Figure 2.2: D2K Naïve Bayes Prediction of Maximum Customer Purchasing Price (MaxPrice) and associated Market Share α_i.

to product design and development using multi-level optimization (analytical target setting [87] and analytical target cascading [72] are utilized in the proposed approach) in the engineering level is illustrated in Figure 2.2.

Commonality is achieved by the linking of component variables among architectures. The proposed methodology suggests that commonality decisions be made on the primary basis of how they affect customer preferences and ultimately, enterprise profit. The Mcconnell/Stigler relationship between unit cost and output suggests that diseconomies of scale may mitigate the cost-savings benefit that commonality provides to the manufacturing process as output increases exponentially [88]. Therefore, the benefits of commonality and modularity will focus less on the manufacturing cost savings, but rather on overall company profit. The reason for this performance metric shift is due to the ambiguities that exist when product manufacturing cost is the primary reason for justifying sharing decisions. Such cost minimization commonality decisions may have adverse effects on the satisfaction of intended customers who may suffer due to performance sacrifices in an attempt to reduce cost. Future research aspirations include incorporating the entire supply chain process into the product family cost model to better understand the effects of downstream processes in enterprise decision making.

2.1.1 Product Portfolio Optimization at Product Family Supersystem Level

The primary product portfolio objective is achieved through a finite launch of product architectures deemed most profitable by the enterprise system level objective. The profit maximizing objective is realized through an iterative process

of acquiring the maximum price (MaxPrice) a customer is willing to pay for a particular product (determined by a customer predictive model), and the cost derived from the component selection process that defines that particular product. The overall maximum profit ($\pi_{overall}$) is used as the metric for this selection process, where $profit_{overall}$ is the summation of the individual product profits that would yield the maximum overall company profit. The product portfolio limit used in this case study is assumed to be the maximum number of product variants in the manufacturing process that would allow the process to still remain efficient, i.e. the point of inflection before a company's manufacturing and distribution capabilities are exceeded.

```
Minimize
```

$$-\sum_{j=1}^{k} x_j \cdot \pi_{Variant(j)} \tag{2.8}$$

```
Where:
```

$\pi_{Variant(j)}$: *Profit of variant (j)*

x_j : *Binary discrete variable selecting or deselecting particular product variant ($\pi_{Variant_j}$)* where $\sum_{j=1}^{k} x_j \leq K$

k**:** *Total feasible product architectures that can be designed*

K**:** *Product portfolio limit (number of architectures in the product family)*

```
Subject To
```

$$h1: \quad x_j = \{0,1\} \quad j \in \{1,\ldots,k\} \tag{2.9}$$

$$g1: \quad \sum_{j=1}^{k} x_j - K \quad \leq 0 \tag{2.10}$$

Product portfolio limit K is a finite number meaning that a company cannot produce every possible combination of architectures, which would be an impractical real life target.

2.1.2 Enterprise System Level

The Naïve Bayes predictive model allows for a customer's maximum price (MaxPrice) value to be used in determining the maximum $Variant_j$ profit ($\pi_{Variant_j}$), where $profit_j$ for a particular product is determined by:

$$\pi_{Variant_{\{j\}}} \quad = \quad MaxPrice_{\{j\}} - Cost_j \tag{2.11}$$

where

- $MaxPrice_j$: is the Naïve Bayesian prediction based on certain input attributes. The MaxPrice can be partitioned during the customer survey to N number of price preference choices to reflect the objective of the enterprise decision maker. $i = [1, ..., N]$. The *most probable* (as defined by equation (2.5)) class variable is used in profit calculation in equation (2.11).
- $Cost_j$: is the engineering sub-level response for the cost needed to produce a product desired by the customers, based on the Naïve Bayes prediction.

The mathematical model at the enterprise level is summarized as follows. (The norm notation indicates $||\cdot|| = ||\cdot||_2^2$, i.e., squared L-2 norm.)

Minimize

$$-\pi_{Variant_j} + \|\mathbf{T}^C - \mathbf{R}^{Ent}\| + \varepsilon_{\mathbf{R}} + \varepsilon_{\mathbf{y}} \tag{2.12}$$

With respect to $\mathbf{R}^{Ent}$=$\mathbf{R}^{Ent}(\mathbf{x}^{Ent}, \mathbf{R}^{Eng})$, $\varepsilon_{\mathbf{R}}$, $\varepsilon_{\mathbf{y}}$

Subject to

$$h1: \quad \pi_{Variant_j} - D \cdot \left(MaxPrice_{\{j\}} - Cost_{\{j\}}\right) = 0 \tag{2.13}$$

$$h2: \quad \sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j} - m = 0 \tag{2.14}$$

$$h3: \quad D - MarketDemand_{Variant_j} = 0 \tag{2.15}$$

$$h4: \quad \sum_{i=1}^{N} \alpha_i - 1 = 0 \tag{2.16}$$

$$g1: \quad \|\mathbf{R}^{Eng} - \mathbf{R}^{Eng^L}\| - \varepsilon_{\mathbf{R}} \leq 0 \tag{2.17}$$

$$g2: \quad \|\mathbf{y} - \mathbf{y}^L\| - \varepsilon_{\mathbf{y}} \leq 0 \tag{2.18}$$

(*h*2: *Given an $m \times n$ matrix of attributes, equality constraint h2 restricts the parameter value of each attribute to only one per row due to the Naïve Bayesian assumption of attribute independence.*)

where:

$\pi_{Variant_j}$: *Profit of product variant j.*

$\mathbf{T}^C$: *Product variant target component predicted by Naïve Bayes customer model.*

$\mathbf{R}^{Ent}$: *Engineering response component cascaded up to the enterprise level.*

$\mathbf{R}^{Ent}$=$\mathbf{R}^{Ent}$($\mathbf{x}^{Ent}$, $\mathbf{R}^{Eng}$) means that the enterprise level response is a function of system variables and the response of the engineering subsystem level.

$\mathbf{y}$: *Linking variables at the enterprise level. The linking variable concept applied to product family design represent shared attributes or components that exist among product variants.*

$\mathbf{y}^{L}$: *Target values for linking variable at the engineering sub-system level cascaded up to enterprise level.*

$\varepsilon_{\mathbf{R}}$: *Deviation tolerance variable between customer component targets and engineering response.*

$\varepsilon_{\mathbf{y}}$: *Deviation tolerance variable between linking variables.*

2.1.3 Engineering Design Subsystem Level

The engineering design sub-system level is defined as the stage in product design wherein engineering design objectives and constraints are formulated to produce a product/variant that satisfies the enterprise level objective [89].

Analytical Target Setting [87]

The multi-objective formulation of the engineering design sublevel focuses primarily on designing an architecture (around which product variants are designed) at product launch that will satisfy customer wants, predicted by the Naïve Bayesian model, while simultaneously minimizing the overall cost of the product. Satisfying customer wants is weighted more due to obvious reasons: a cheaper product will not automatically translate into an attractive product if customer preferences are not satisfied. The mathematical model at the engineering design level is summarized as follows:

```
Minimize
```

$$Cost_j + \|\mathbf{R}^{Eng^U} - \mathbf{R}^{Eng}\| + \|\mathbf{y}_j - \mathbf{y}_j^U\| \tag{2.19}$$

$Cost_j$: *Cost of product variant (j)*

$\mathbf{R}^{Eng^U}$: Response from the enterprise system level, cascaded down to the engineering level (At the enterprise system level mathematical formulation, $\mathbf{R}^{Eng^U}$ is simply $\mathbf{R}^{Eng}$).

$\mathbf{R}^{Eng}$: Response from the engineering sublevel, i.e. $\mathbf{R}^{Eng}$=$\mathbf{R}^{Eng}$($\mathbf{x}_{Eng}$, $\mathbf{y}_{Eng}$). It means that the response of the engineering design sublevel is a function of local design variables and also sharing linking variables ($\mathbf{R}^{Eng}$ at the engineering subsystem level will become $\mathbf{R}^{Eng^L}$ at the enterprise system level).

$\mathbf{y}_j$: Linking variables at the engineering design level

$\mathbf{y}_j^U$: Target value for linking variables from the upper enterprise level cascaded down to engineering level

Subject To

$$h1: \quad Cost - \sum_{i=1}^{J} x_i q_i = 0 \tag{2.20}$$

$$\text{design constraints} : \mathbf{g}_j(\mathbf{x}_{Eng}) \leq \mathbf{0} \tag{2.21}$$

x_i : Binary discrete variable selecting or deselecting particular product variant component ($q_{Variant}$)

q_i : Product variant Component (Discrete or Continuous Variable)

- *Discrete Component Variable:* Manufactured by a supplier with predefined performance and cost attributes
- *Continuous Component Variable:* Company manufactured with changing specifications to cater to dynamic architecture design

J : Total available components in the engineering product design

The engineering design sub-system level objective is modeled as a mixed integer nonlinear programming problem with discrete variables that dictate the component selection process and continuous variables for the engineering designed components. A branch and bound algorithm is used to achieve an optimal solution [86]. Since there are both discrete and continuous variables in the mathematical model, the branch and bound algorithm attempts to find an optimal solution by first solving the relaxation problem (i.e., integer restrictions are relaxed) which is simply a nonlinear optimization problem [86]. In the subsequent solution, if all the discrete variables take integer values, then the mixed integer programming problem is solved and an optimal solution is reached [86]. For each discrete variable that does not take on an integer value, the algorithm takes this variable and divides the problem (branches) into two new nonlinear programming problems. This process is continued until a global optimum is achieved.

2.2 Application

2.2.1 Product Portfolio Formulation: Cellular Phone Product Family

To demonstrate the effectiveness of the proposed methodology, a realistic cell phone case study is presented. The methodology begins by introducing a customer survey questionnaire (Table 2.3) that is modeled to realistically capture the true essence of what customers want. The data is acquired through a realistic customer survey, although the

Table 2.3: Customer Survey Questions and Response Options

Customer Data Acquisition Process

Survey Questions	Survey Answer Choices
What feature would you most like your cellular phone to have?	MP3, Camera, Internet, Games, SMSText, Just Talk
What is more important to you?	Weight, Size , or Cost of Cell Phone
What type of Cell Phone Design do you Like?	Flip Phone Design, Shell Phone Design
What Type of Connectivity would you prefer your phone to have?	Bluetooth, WiFi, Infrared, None
What is the minimum talk time you require before a recharge?	3 Hours, 5 Hours, 7 Hours
What is more important to you?	Display Screen Size, Display Resolution
What is the Maximum Price you would be willing to pay for the features you just described?	$40, $80, $120, $160, $200

proposed framework is not limited to customer survey methods. For example, data can also be acquired from existing company databases containing transactional data. Performance metrics determined by the customer prediction will be set as targets at the engineering level.

From the results of the survey, a model can now be developed that will predict the maximum purchasing price that a customer is willing to pay based on certain attributes. The D2K software helps to develop this model with the transformation of the customer raw data. MaxPrice can then be used in a sensitivity analysis to determine the profit for a particular product design, given certain selected attributes. The attributes selected are used as targets in the engineering level.

2.2.2 Enterprise System Level

Cell Phone System Profit Optimization

The Naïve Bayesian model developed by D2K allows the user to select/deselect attributes and observe the change in MaxPrice and the market share associated with each. As can be seen in figure 2.2, an MP3 phone architecture is used as a starting point with customer attributes including **[5 hour Battery life [90], Cost Objective, Screen Size Priority, Wifi Connectivity [91], and Shell Phone Design]** .

These attribute targets are then cascaded down to the engineering sub-level to determine whether or not such a product design is feasible. The MaxPrice prediction is used at the enterprise level in calculating the Profit(π) for this

particular MP3 Phone. Mathematically, this is represented as:

Minimize

$$-\pi_{MP3Variant_1} + \|T^{BatteryLife} - R^{BatteryLife^{Ent}}\| + \|T^{Wifi} - R^{Wifi^{Ent}}\| + \|T^{Shell} - R^{Shell^{Ent}}\| + \varepsilon_{BatteryLife} + \varepsilon_{Wifi} + \varepsilon_{Shell} \tag{2.22}$$

In the cell phone case study, $R^{BatteryLife^{Ent}}$ is considered as a linking variable at the engineering design level. Thus, a deviation constraint $g1$ is added in the constraint set.

Subject To

$$h1: \quad \pi_{MP3Variant_1} - D \cdot (MaxPrice_{\{MP3_1\}} - Cost_{\{MP3_1\}}) = 0 \tag{2.23}$$

$$h2: \quad MaxPrice_{\{MP3_1\}} - \$200 = 0 \tag{2.24}$$

$$h3: \quad \sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j} - m = 0 \tag{2.25}$$

$$h4: \quad \sum_{i=1}^{K} \alpha_i - 1 = 0 \tag{2.26}$$

$$h5: \quad \alpha_i = \{51\%, 23\%, 22\%, 3\%, 1\%\} \tag{2.27}$$

$$h6: \quad MaxPrice_i = \{200, 120, 160, 80, 40\} \tag{2.28}$$

$$g1: \quad \|R^{BatteryLife^{Ent}} - R^{BatteryLife^{Eng^L}}\| - \varepsilon_{BatteryLife} \leq 0 \tag{2.29}$$

$$g2: \quad \|R^{Wifi^{Ent}} - R^{Wifi^{Eng^L}}\| - \varepsilon_{Wifi} \leq 0 \tag{2.30}$$

$$g3: \quad \|R^{Shell^{Ent}} - R^{Shell^{Eng^L}}\| - \varepsilon_{Shell} \leq 0 \tag{2.31}$$

$$g4: \quad D - D_0 \leq 0 \tag{2.32}$$

where:

$\pi_{MP3Variant_j}$: *Profit (in $) of MP3 Variant with specific design features*

$\mathbf{T}^{BatteryLife}$: *Battery Life (hours) Target predicted by Customer Naïve Bayes Model*

$\mathbf{R}^{BatteryLife}$ **:=Battery Life Response cascaded up from Engineering Sublevel**

$\mathbf{T}^{Wifi}$: *Connectivity (bluetooth,wifi,or infrared) Target predicted by Customer Naïve Bayes Model*

$\mathbf{R}^{Wifi}$:=**Connectivity Response cascaded up from Engineering Sublevel**

$\mathbf{T}^{Shell}$: *Shell Design Target predicted by Customer Naïve Bayes Model*

$\mathbf{R}^{Shell}$**=Shell Design Response cascaded up from Engineering Sublevel**

$\mathbf{y}$: *Linking variables at the enterprise level. The linking variable concept applied to product family design represents shared attributes or components that exist among product variants.*

$\mathbf{y}^{L}$: *Target values for linking variable at the engineering sub-system level cascaded up to enterprise level.*

$\varepsilon_{\textbf{Battery Life}}$: *Deviation tolerance variable between customer component targets and engineering response.*

$\varepsilon_{\textbf{Wifi}}$: *Deviation tolerance variable between customer component targets and engineering response.*

$\varepsilon_{\textbf{Shell}}$: *Deviation tolerance variable between customer component targets and engineering response.*

Here, D_0 **= 100,000** (Represents the total market population of cell phone consumers) and K is 5 (Product Portfolio limit that would enable manufacturing process to remain efficient). The table of demand information for a given class variable prediction is given in Table 2.4, where **D=8395** represents the demand for a $200 phone based on the Naïve Bayesian model in equation (2.5). One of the values of MaxPrice will be selected for each attribute combination (customer predicted preference) so long as it is *more probable* than any other class variable of MaxPrice. (See Equation (2.5))

Table 2.4: Demand information based on the Naïve Bayesian predictive model

MaxPrice	Customer demand (D) at given price
$200	8,395
$160	16,001
$120	21,796
$80	12,908
$40	7,899

2.2.3 Engineering Sub-system Level

Cell Phone Subsystem

The engineering sub-system level comprises of a multi-objective function of cost minimization while simultaneously minimizing the deviation between customer design targets and engineering response. During the first iteration in the

Product Portfolio optimization, linking variables are nonexistent due to the fact that only one optimal cell phone design exists in the product portfolio set.

The basic mathematical formulation of successive cell phone variants is similar to perturbations occurring with each successive product variant (Figure 2.2), depending on the customer targets.

Minimize

$$Cost_{MP3Variant_1} + \|R^{BatteryLife^U} - R^{BatteryLife}\| + \|R^{Wifi^U} - R^{Wifi}\| + \|R^{Shell^U} - R^{Shell}\| \tag{2.33}$$

Subject To:

Screen Resolution Constraints

$$h1: \quad (A1 * LCD_{length} * LCD_{width}) - LCD_{res} = 0 \tag{2.34}$$

$$h2: \quad (A2 * LCD_{length} * LCD_{width}) - Cost_{LCD} = 0 \tag{2.35}$$

$$h3: \quad (A3 * LCD_{length} * LCD_{width}) - Weight_{LCD} = 0 \tag{2.36}$$

$$h4: \quad (A4 * LCD_{length} * LCD_{width}) - Power_{LCD} = 0 \tag{2.37}$$

$$h5: \quad (A5 * OLED_{length} * OLED_{width}) - OLED_{res} = 0 \tag{2.38}$$

$$h6: \quad (A6 * OLED_{length} * OLED_{width}) - Cost_{OLED} = 0 \tag{2.39}$$

$$h7: \quad (A7 * OLED_{length} * OLED_{width}) - Weight_{OLED} = 0 \tag{2.40}$$

$$h8: \quad (A8 * OLED_{length} * OLED_{width}) - Power_{OLED} = 0 \tag{2.41}$$

Battery Design Constraints

$$h9: Cap_{NIMH} - ((NIMH_{Const1} * (V_{NIMH})) - T_{Hours} * \sum_{i=1}^{N} P_{Component_i} = 0 \tag{2.42}$$

$$h10: Cap_{LION} - ((LION_{Const1} * (V_{LION})) - T_{Hours} * \sum_{i=1}^{N} P_{Component_i} = 0 \tag{2.43}$$

$$h11: Battery_{TalkTime} - (NIMH * ((0.0053 * (Capacity_{NIMH})) + 0.0269) + (LION * ((0.0061 * (Capacity_{LION})) + 0.1667))) = 0 \tag{2.44}$$

$$h12: \quad ((NIMH_{Const2}*(L_{NIMH}*W_{NIMH}*T_{NIMH}))-Cost_{NIMH})=0 \tag{2.45}$$

$$h13: \quad (LION_{Const2}*(L_{LION}*W_{LION}*T_{LION}))-Cost_{LION})=0 \tag{2.46}$$

$$h14: \quad ((NIMH_{Const3}*(L_{NIMH}*W_{NIMH}*T_{NIMH}))-Wg_{NIMH})=0 \tag{2.47}$$

$$h15: \quad ((LION_{Const3}*(L_{LION}*W_{LION}*T_{LION}))-Wg_{LION})=0 \tag{2.48}$$

$$g1: \quad (NIMH*L_{NIMH}+LION*L_{LION})-0.60 *(SHELL*L_{SHELL}+FLIP*L_{SHELL})\leq 0 \tag{2.49}$$

$$g2: \quad (NIMH*W_{NIMH}+LION*W_{LION}) -0.95*(SHELL*W_{SHELL}+FLIP*W_{FLIP})\leq 0 \tag{2.50}$$

$$g3: \quad (NIMH*T_{NIMH}+LION*T_{LION}) -0.45*(SHELL*T_{SHELL}+FLIP*T_{FLIP})\leq 0 \tag{2.51}$$

(To enhance the overall flow of the dissertation, several variable names are abbreviated (L=Length, W=Width, T=Thickness, Wg=Weight, V=Volume, Cap=Capacity, P=Power Consumption, etc.))

Cell Phone Outer Casing Design Constraints

$$h16: \quad (SHELL_{Const1}*L_{SHELL}*W_{SHELL}*T_{SHELL})-Cost_{SHELL}=0 \tag{2.52}$$

$$h17: \quad (FLIP_{Const1}*L_{FLIP}*W_{FLIP}*T_{FLIP})-Cost_{FLIP}=0 \tag{2.53}$$

$$h18: \quad (SHELL_{Const2}*L_{SHELL}*W_{SHELL}*T_{SHELL})-Wg_{SHELL}=0 \tag{2.54}$$

$$h19: \quad (FLIP_{Const2}*L_{FLIP}*W_{FLIP}*T_{FLIP})-Wg_{FLIP}=0 \tag{2.55}$$

$$g4: \quad L_{LCD}-(0.60*SHELL*L_{SHELL}+0.60*FLIP*L_{FLIP})\leq 0 \tag{2.56}$$

$$g5: \quad (0.30*SHELL*L_{SHELL}+0.30*FLIP*L_{FLIP})-L_{LCD}\leq 0 \tag{2.57}$$

$$g6: \quad W_{LCD}-0.90*(SHELL*W_{SHELL}+FLIP*W_{FLIP})\leq 0 \tag{2.58}$$

$$g7: \quad 0.7*(SHELL*W_{SHELL}+FLIP*W_{FLIP})-W_{LCD}\leq 0 \tag{2.59}$$

$$g8: \quad L_{OLED}-(0.60*SHELL*L_{SHELL}+0.60*FLIP*L_{FLIP})\leq 0 \tag{2.60}$$

$$g9: \quad (0.30*SHELL*L_{SHELL}+0.30*FLIP*L_{FLIP})-L_{OLED}\leq 0 \tag{2.61}$$

$$g10: \quad W_{OLED}-0.90*(SHELL*W_{SHELL}+FLIP*W_{FLIP})\leq 0 \tag{2.62}$$

$$g11: \quad 0.7*(_{SHELL*Width_{SHELL}}+FLIP*Width_{FLIP})-_{OLED_{width}}\leq 0 \tag{2.63}$$

Table 2.5: Possible Shared Component Variables

Component	Description	Design Options
Internal Memory (RAM)	32 MB RAM Discrete Choice Variable	Manufacturer
Internal Memory (RAM)	64 MB RAM Discrete Choice Variable	Manufacturer
External Memory	Memory Stick Pro Discrete Choice Variable	Manufacturer
External Memory	Memory Stick Duo Discrete Choice Variable	Manufacturer
Hard Drive	1GB Storage Discrete Choice Variable	Manufacturer
Hard Drive	2GB Storage Discrete Choice Variable	Manufacturer
Phone Design	Shell Phone Design Variables	EngineeringDesign
Phone Design	Flip Phone Design Variables	EngineeringDesign
Battery Type	Lithium Polymer [5] Battery Design Variables	EngineeringDesign
Battery Type	Lithium Ion [5] Battery Design Variables	EngineeringDesign
Connectivity	Bluetooth Connection Discrete Variable	Manufacturer
Connectivity	Wifi Discrete Choice Variable	Manufacturer
Connectivity	Infra Red Discrete Choice Variable	Manufacturer
Audio Codec	Microphone Discrete Variable	Manufacturer
Audio Codec	Earpiece Discrete Variable	Manufacturer
Audio Codec	Audio Jack Discrete Variable	Manufacturer
Audio Codec	External Speaker Discrete Variable	Manufacturer
Display Type	TFT LCD [24] Discrete Variable	Manufacturer
Display Type	OLED [24]Discrete Variable	Manufacturer

Design Objectives

$$h20: \quad TotalCost - \sum_{i=1}^{N} Component(i)_{Cost} = 0 \tag{2.64}$$

$$h21: \quad TotalWeight - \sum_{i=1}^{N} Component(i)_{Weight} = 0 \tag{2.65}$$

Table 2.5 identifies the possible shared components of each individual MP3 capable phone. Sharing decisions are influenced by customer performance expectations and engineering capabilities. Certain components are purchased directly from a manufacturer and would therefore have fixed performance specifications, while other components can be manufactured by the company to meet customer needs.

2.2.4 Optimization Study

With a methodological approach to product architecture formulation, enterprise decision makers can have a validation tool to justify product portfolio formulation and launch decisions. The study begins by defining a finite number of unique architectures that will constitute the optimal product family. For the study, a product portfolio of five product variants will be set as the maximum manufacturing ability. It is assumed in this case study that a product portfolio greater than five will begin to result in diseconomies of scale and ultimately, reduced profit [88].

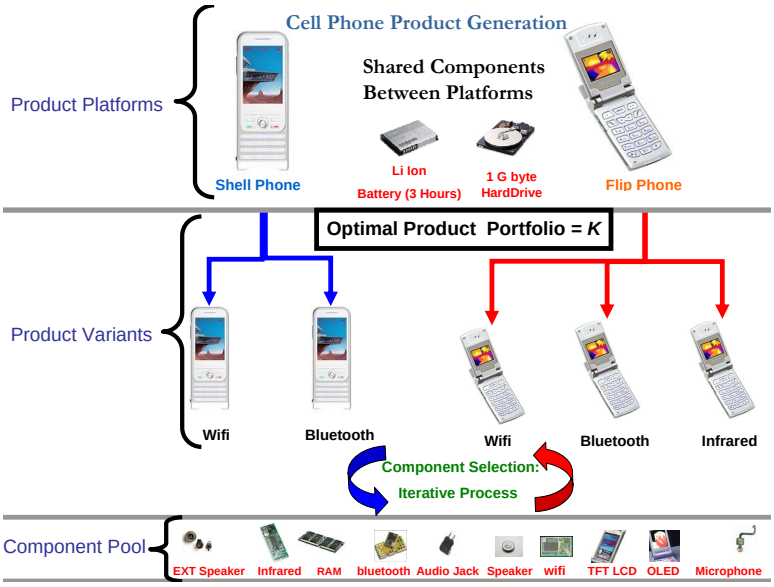


Figure 2.3: Optimal Product Portfolio Example. Illustrates how just two product architectures can generate product variants that make up a family of products (product portfolio of $K = 5$ products).

Predictive product performance targets are acquired through the proposed data mining process and used to set the initial starting point values at the enterprise level as targets for the engineering subsystem level, i.e. **[MP3, 5 hour Battery Life, Cost Objective, Screensize, Wifi Connectivity, Shell Design]**. For this particular Cell Phone design, an engineering product design cost of \$98.3/unit is attained. The cost response from the engineering design level is then cascaded up to the enterprise level where the *MaxPrice* is used to calculate the predicted profit ($\pi_{Variant}$). With a predicted MaxPrice of \$200 and an associated demand (D) of 8395, the resulting projected profit is \$853,448 . The particular product design however fails to meet the battery life target of 5 hours, instead designing a cell phone with a battery life of only 4.5 hours.

The customer focused objective of matching predicted performance expectations and the engineering design objective of designing the lowest cost product are competing objectives. The enterprise profit calculations presented in this work are the projected profit calculations for a given product launch, based on a particular phone design and how closely it matches customer performance expectations (which may consequently affect the product demand). The argument is made that it is better to launch a product that has lower projected profits (but fully satisfies customer expectations) than to launch a product with a higher projected profit margin (but fails to satisfy customer wants). Failure to satisfy customer wants would adversely affect the *actual* demand for that product and decrease the actual enterprise profit as customers switch to alternative products that more closely satisfy their performance expectations. The disparity between *projected* vs. *actual* profit calculations is therefore highly dependent on product performance.

Optimal Product Portfolio Model

Mathematically, this optimization process is translated to

`Minimize`

$$\sum_{j=1}^{k} -x_j(D_j \cdot (\pi_{Variant(j)} - \lambda \cdot \sum_{i=0}^{N} \varepsilon_i)) \tag{2.66}$$

$\pi_{Variant(j)}$: *Profit for Variant (j)*

x_j : *Binary discrete variable selecting or deselecting particular product variant* ($\pi_{Variant(j)}$) where $\sum_{j=1}^{k} x_j \leq K$, where K is the optimal product portfolio limit of 5 for the example and k symbolizes the 9 studied MP3 architectures.

D_j**:** *Demand for a given product design*

λ**:** *Weighted value for penalty term* ε_i

ε_i**:** *Tolerance deviation term for particular customer target (i)*

N**:** *Total number of shared components*

The comparative analysis can now begin where the above calculated profit ($853,448) is used as a reference amount. The profits of the first five product variants that can be feasibly designed with the MP3 technology are calculated. For each successive iteration, the newly calculated profit of $Variant_j$ will be compared to that of each $Variant_{i[i=1,\ldots,K]}$ existing in the feasible product family set. If the newly calculated product variant profit is greater than any of the variant profits in the set, the least profitable product variant is discarded and replaced with $Variant_j$.

Depending on the number of possible combinations of the predictive model, either an exhaustive search approach or a tree branching algorithm can be used. For the MP3 Architecture, nine combinations are analyzed with a battery life of 3 hours as the primary sharing component.

2.3 Results and Discussion

To determine the optimal product portfolio of five architectures, the nine combinations of MP3 capable cell phones were analyzed (Tables 2.6 and 2.7) to determine the profit margins of each product variant. The optimization results reveal that $Variant_1$ fails to satisfy customer targets on one of the performance metrics. I.e. deviation between customer battery life target and engineering battery manufacturing capabilities $\|T^{BatteryLife} - R^{BatteryLife}\|$ is greater than tolerance $\varepsilon_{BatteryLife}$ and is therefore deemed less profitable with the incorporation of the penalty term described in equation (2.66).

Table 2.6: Optimal Product Family Results (Part 1). Highlighted architectures represent the most profitable product portfolio given maximum five architectures allowed in the portfolio.

Variable Description	Component Source	MP3 Phone1 Solution	MP3 Phone2 Solution	MP3 Phone3 Solution	MP3 Phone4 Solution	MP3 Phone5 Solution	MP3 Phone6 Solution	MP3 Phone7 Solution	MP3 Phone8 Solution	MP3 Phone9 Solution	Units
32 Megabyte Discrete Variable	Manufacturer 1	0	1	1	1	1	1	1	1	1	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	-
64 Megabyte Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	1	0	0	0	0	0	0	0	0	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	-
Memory Stick Pro Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	
Memory Stick Duo Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	-
1GB Storage Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	1	1	1	1	1	1	1	1	1	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	-
2 GB Storage Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	
Shell Phone Discrete Variable	Engineering Design	1	1	0	1	0	1	0	1	0	-
Phone Length	Engineering Design	85.0	80.0	120.0	80.0	120.0	80.0	120.0	80.0	120.0	mm
Phone Width	Engineering Design	48.4	40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	mm
Phone Thickness	Engineering Design	19.1	17.7	16.3	17.7	16.3	16.6	12.0	17.4	12.0	mm
Phone Weight	Engineering Design	40.0	28.9	40.0	28.9	40.0	27.0	29.4	28.4	29.4	g
Phone Cost	Engineering Design	18.0	13.0	18.0	13.0	18.0	12.1	13.2	12.8	13.2	$
Flip Phone Discrete Variable	Engineering Design	0	0	1	0	1	0	1	0	1	-
Phone Length	Engineering Design	100.0	100.0	100.0	100.0	100.0	135.1	100.0	100.0	100.0	mm
Phone Width	Engineering Design	45.0	68.0	45.0	68.0	45.0	45.0	45.0	45.0	45.0	mm
Phone Thickness	Engineering Design	18.1	12.0	12.0	12.0	12.0	12.0	12.0	18.1	12.0	mm
Phone Weight	Engineering Design	40.0	40.0	26.5	40.0	26.5	35.7	26.5	40.0	26.5	g
Phone Cost	Engineering Design	12.0	12.0	7.9	12.0	7.9	10.7	7.9	12.0	7.9	$
Nickel Metal Hydride Battery Discrete Variable	Engineering Design	0	0	0	0	0	0	0	0	0	-
Battery Weight	Engineering Design	42.4	49.2	50.0	50.0	50.0	50.0	50.0	50.0	50.0	g
Battery Length	Engineering Design	46.2	51.1	36.6	59.7	72.5	38.2	80.0	70.0	80.0	mm
Battery Width	Engineering Design	54.7	34.2	48.6	37.0	32.3	60.0	60.0	50.4	25.5	mm
Battery Thickness	Engineering Design	17.9	30.0	30.0	24.1	22.7	23.3	11.1	15.1	26.1	mm
Battery Cost	Engineering Design	17.1	19.9	20.2	20.2	20.2	20.2	20.2	20.2	20.2	$
Battery Capacity	Engineering Design	803.9	958.9	962.6	976.8	962.6	1016.8	1002.6	986.8	972.6	mAh
Lithium Ion Battery Discrete Variable	Engineering Design	1	1	1	1	1	1	1	1	1	-
Battery Weight	Engineering Design	17.76	12.82	13.11	12.82	13.11	12.00	12.29	12.62	12.91	g
Battery Length	Engineering Design	51.00	48.00	72.00	48.00	72.00	48.00	72.00	48.00	63.58	mm
Battery Width	Engineering Design	45.97	38.00	42.75	38.00	42.75	38.00	42.75	38.00	42.74	mm
Battery Thickness	Engineering Design	8.58	7.96	4.82	7.96	4.82	7.45	4.52	7.83	5.38	mm
Battery Cost	Engineering Design	16.16	11.67	11.93	11.67	11.93	10.92	11.19	11.48	11.74	$
Battery Capacity	Engineering Design	706.77	464.47	464.47	464.47	464.47	464.47	464.47	464.47	464.47	mAh
Cell Phone Talk Time	Engineering Design	4.48	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	h
Bluetooth Discrete Variable	Manufacturer 1	0	0	0	1	1	0	0	0	0	-
WiFi Discrete Variable	Manufacturer 1	1	1	1	0	0	0	0	0	0	-
Infra-Red Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	1	1	-

Table 2.7: Optimal Product Family Results (Part 2). Highlighted architectures represent the most profitable product portfolio given maximum five architectures allowed in the portfolio.

Variable Description	Component Source	MP3 Phone1 Solution	MP3 Phone2 Solution	MP3 Phone3 Solution	MP3 Phone4 Solution	MP3 Phone5 Solution	MP3 Phone6 Solution	MP3 Phone7 Solution	MP3 Phone8 Solution	MP3 Phone9 Solution	Units
Microphone Discrete Variable	Manufacturer 1	1	1	1	1	1	1	1	1	1	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
EarPiece Discrete Variable	Manufacturer 1	0	0	1	0	1	0	1	0	1	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
	Manufacturer 3	1	1	0	1	0	1	0	1	0	-
Audio Jack Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
External Speaker Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	1	0	0	0	0	0	0	0	0	-
	Manufacturer 3	0	1	1	1	1	1	1	1	1	-
LCD Discrete Variable	Engineering Design	1	1	1	1	1	1	1	1	1	-
Length of LCD	Engineering Design	25.50	24.00	30.00	24.00	30.00	24.00	30.00	24.00	30.00	mm
Width of LCD	Engineering Design	34.94	28.00	31.50	28.00	31.50	28.00	31.50	28.00	31.50	mm
Display Resolution	Engineering Design	13131.32	9905.28	13929.30	9905.28	13929.30	9905.28	13929.30	9905.28	13929.30	pixels
LCD Manufacturing Cost	Engineering Design	4.45	3.36	4.73	3.36	4.73	3.36	4.73	3.36	4.73	$
LCD Unit Weight	Engineering Design	35.63	26.88	37.80	26.88	37.80	26.88	37.80	26.88	37.80	g
LCD Power Consumption	Engineering Design	8.91	6.72	9.45	6.72	9.45	6.72	9.45	6.72	9.45	mhA
OLED Discrete Variable	Engineering Design	0	0	0	0	0	0	0	0	0	-
Length of OLED	Engineering Design	25.50	35.33	30.00	25.57	30.00	33.41	31.75	33.21	30.00	mm
Width of OLED	Engineering Design	39.22	28.09	31.50	36.00	31.50	28.00	31.50	28.00	33.33	mm
Display Resolution	Engineering Design	19620.00	19473.58	18540.90	18060.00	18540.90	18352.50	19620.00	18245.00	19620.00	pixels
OLED Manufacturing Cost	Engineering Design	8.00	7.94	7.56	7.36	7.56	7.48	8.00	7.44	8.00	$
OLED Unit Weight	Engineering Design	30.00	29.78	28.35	27.61	28.35	28.06	30.00	27.90	30.00	g
OLED Power Consumption	Engineering Design	30.00	29.78	28.35	27.61	28.35	28.06	30.00	27.90	30.00	MhA
1 MegaPixel Camera Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
2 MegaPixel Camera Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
MP3 Module Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	1	1	1	1	1	1	1	1	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	-
	Manufacturer 4	1	0	0	0	0	0	0	0	0	-
Internet Module Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
Graphics Module for Games: Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	-
SMS Text Message Technology: Discrete Variable	Manufacturer 1	0	0	0	0	0	0	0	0	0	-
Total Architecture Cost	**Engineering Solution**	98.34	84.46	81.05	82.96	79.55	75.59	73.01	80.50	77.30	$/unit
Total Architecture Weight	**Engineering Solution**	141.82	121.29	129.69	117.39	125.79	105.83	116.07	116.33	125.19	g

Each subsequent product variant uses a battery life of 3 hours, which may initially seem less profitable due to changes in the Naïve Bayes predictions of MaxPrice (Figure 2.2 and Figure 2.3). After an engineering design validation however, it can be seen that such architectures would be cheaper to manufacture and would yield the highest profit margins while satisfying customer wants. (**note:** An engineering response is an evaluation of technical capabilities by the engineering team in determining the feasibility of such a product. The relevance to battery life is that certain product concepts may have unattainable battery life expectations.)

The optimal product portfolio (Figure 1.2) given this approach will therefore be architectures $\{3,5,6,7,9\}$, yielding a total company profit of:

$$\begin{aligned}\pi_{Overall} &= \$848,902 + \$881,596 + \$967,996 \\ &+ \$1,024,232 + \$930,769 = \textbf{\$4.65 Million}\end{aligned} \tag{2.67}$$

The multi-level optimization solution (adopting the ATC methodology [72]) took approximately 500 seconds per product variant running on an Intel Pentium IV desktop (3.2 GHz). The model was developed in the Matlab® [92] environment with Tomlab® [86] used in the optimization sequence.

The cost-savings benefit of manufacturing can be realized when a product manufacturing process has minimal number of interruptions. Thus the more components that a product shares with variants, the higher the probability of lowering overall company operating costs. Sharing decisions focused solely on manufacturing process cost savings can however have adverse effects on customer preferences and ultimately their willingness to pay as seen in the following example. Four out of the nine product architectures share a **flip phone design** (Table 2.6, 2.7). Although it would be more desirable for all architectures to share the same type of design (flip or shell), it is clearly observed that such a decision would not yield the most profitable product portfolio. For example, sharing a **shell phone design** for 5 architectures would mean selecting architectures $\{1,2,4,6,8\}$, which would yield a maximum profit of:

$$\begin{aligned}\pi_{Overall} &= \$853,448 + \$774,642 + \$807,336 \\ &+ \$967,996 + \$632,093 = \textbf{\$4.03 Million}\end{aligned} \tag{2.68}$$

Even without penalizing $Variant_1$ for failing to satisfy the customer battery target of 5 hours (*actual engineering response =4.5 hours*), a sharing decision of a SHELL phone design would yield a less profitable product portfolio. The solution to product portfolio optimization is multifaceted requiring input from different specializations across different boundaries. Such powerful insights will help enterprise decision makers understand the intricate link that exists between what customer wants and engineering design capabilities.

2.4 Conclusion

The emergence of a customer driven need for product differentiation has lead companies to re-evaluate current design and manufacturing processes. Consequently, analytical techniques are required to alleviate the costs associated with product differentiation. The greatest challenge is to develop an optimal product architecture for a family of products in a dynamic market space. To overcome this challenge, the proposed methodology has successfully demonstrated how data mining techniques can help develop a product family by encompassing customer requirements directly with engineering capabilities using ATS [87] and ATC [89]. Modularity and component sharing decisions can now be expanded beyond manufacturing cost savings to include consumer price sensitivity to product architecture changes. The dynamic product architecture concept utilized in this work has the benefit of continuously changing architecture design variables throughout the product design phase to cater to customer preference requirements. A product portfolio is achieved which not only maximizes profit, but simultaneously satisfies what the customer wants.

The cell phone analysis systematically attains a feasible product portfolio by simultaneously focusing on changing demand due to a particular product design choice. The model places emphasis on deterministic (and in later works stochastic) methods in product architecture formulation. The long term goal is to provide decision makers in industry with a useful tool that helps mitigate the associated risks involved in product portfolio formation and product launch decisions. Such a tool has the potential to drastically reduce errors associated with ad hoc product portfolio methodologies or disjointed expertise between the business and engineering teams. The manufacturing benefits of product architecture design and product portfolio formulation will be incorporated in later models to reflect a wider scope of product design. Careful attention will be paid to the efficiency at which the algorithm of choice will converge to an optimal solution. An exhaustive search algorithm or a branch and bound algorithm will help to ensure a global optimum for the maximum attainable profit for a family of architectures.

Chapter 3

ReliefF and X-Means Clustering Product Family Methodology

3.1 Product Family Design Methodology

The primary contribution of this work is to present a product family design methodology for complex engineering systems that *autonomously identifies the number of products to design* by extracting *weighted product preference information* from a customer data set. This work focuses on design problems with large product preference data sets that can be integrated into the product design process. Figure 3.1 represents the overall methodology that begins with the acquisition of raw customer product preference data and employs data mining attribute weighting and clustering techniques to determine the number of unique products needed for a given data set. For this, the ReliefF attribute weighting algorithm is employed to identify attributes in order of importance in the data set. Then, the X-means clustering algorithm is employed to identify groups of similar operating states within the raw data set. As a result, the number of product variants that should be introduced to reflect preferences (represented in the data) can be identified. Steps 2 and 3 in Figure 3.1 illustrate the added benefits of component sharing by clustering similar products together in an attempt to reduce product design costs. The X-Means clustering technique is employed at the engineering design level to determine which products are similar enough to potentially benefit from component sharing decisions. These sharing design decisions are implemented in a multidisciplinary design optimization framework where an individual product variant is modeled as an individual subsystem. For products with highly diverse operating conditions, the data set itself may be highly heterogeneous, making it quite difficult for engineers to determine the number of products to design in order to satisfy the market space. By employing the Data Mining ReliefF attribute weighting and X-Means clustering techniques to the raw data set (Figure 3.1), engineers can determine the initial product architectures to design. The details of Figure 3.1 will now be explained in depth in the following sections.

3.1.1 Data Mining Product Preferences

The Data Mining of product preferences is the stage where dominant patterns are identified within the raw data set [54]. With each unique instance in the data set representing a customer's preferred operating state for the product, the number of operating states can increase rapidly, thereby making it impractical for a single design to exist for

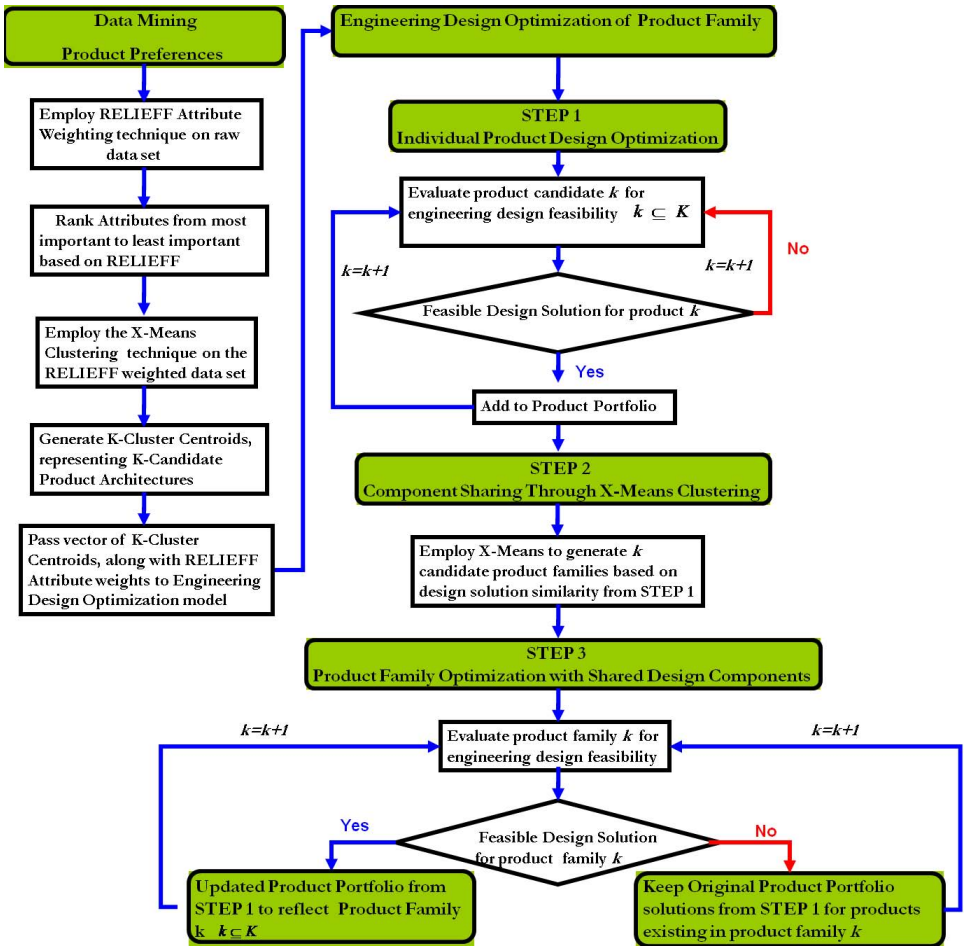


Figure 3.1: Flow Diagram of Proposed Product Family Design Methodology: From Data Mining Product Architecture Identification to Component Sharing through X-Means Clustering.

each unique state. The engineering design goal is to identify operating states within the data set that are *similar* in design requirements (as determined by the data mining algorithm). Since product attributes may vary in terms of design significance, an appropriate attribute weighting technique would help guide the engineering design process. To accomplish these product design challenges, the ReliefF attribute weighting algorithm is first employed to weight attributes in order of importance in the data set [93]. The X-means clustering algorithm is then employed to identify groups of similar operating states within the raw data set (illustrated in the Data Mining flow diagram in Figure 3.1 and visually represented on the left in Figure 2). The weighted attributes will influence both the data clustering process as well as the engineering design model and more valued product attributes will be given more weight in the overall product family design methodology. The X-means clustering algorithm is employed again in the component sharing decisions during the product family optimization stage as similar individual product designs are grouped together by similarity of design (Steps 2 and 3 in Figure 3.1 and visually represented on the right in Figure 3.2). Below is an introduction to the ReliefF attribute weighting technique that will later be applied to the raw data set.

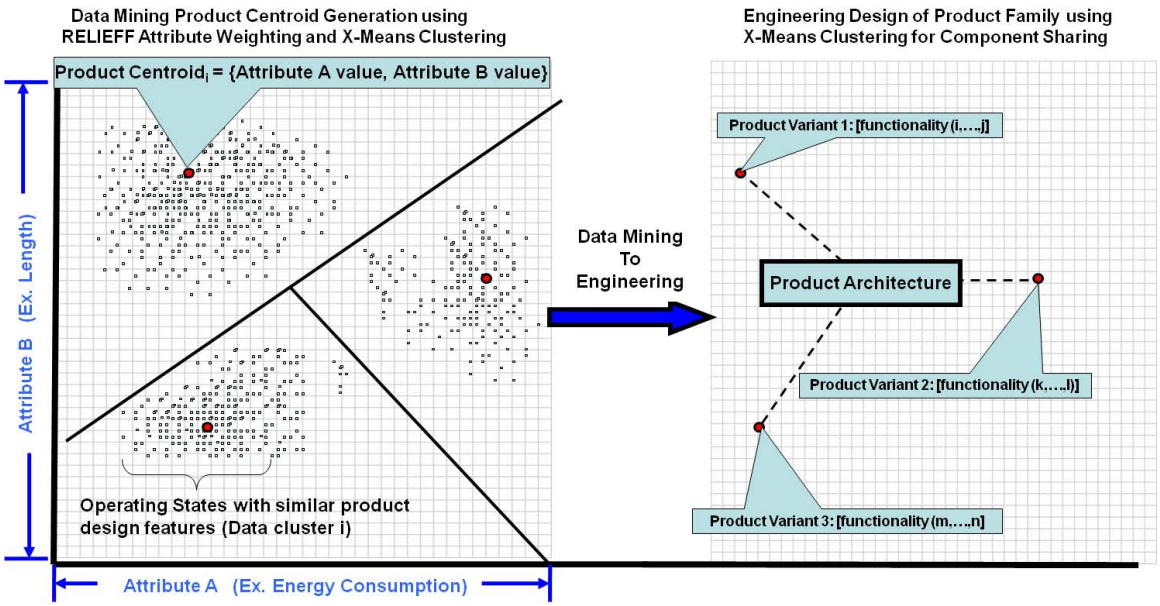


Figure 3.2: Visual representation of family design based on Data Mining ReliefF attribute weighting and X-means clustering for Product Centroid generation and X-Means Clustering for Product Family Component Sharing Optimization.

ReliefF Product Attribute Weighting Algorithm

In this work, the enhanced version of the *Relief* algorithm called ReliefF is employed [94]. ReliefF extends the original Relief algorithm by enabling it to efficiently handle *multi-class variables* and also missing values within the data set [94]. A *class variable* can be thought of as the response or predictor variable of interest. Examples of class variables in product design data sets may include efficiency, energy consumption, price, etc.

The original Relief algorithm proposed by Kira and Rendell is an attribute evaluation technique that will enable product development engineers to extract the importance of individual product attributes within a raw data set without explicit user provided ranking information [93]. This can prove to be a vital time saving strategy, especially for extremely large data sets containing many attributes. Identifying the order of attribute relevance within a data set can reduce the overall computational complexity and increase the efficiency of data mining algorithms [93, 94].

Given a raw data set S, m instances are selected to serve as the number of sampled instances where p denotes the total unique attributes within the sample set m [93]. The overall objective of the Relief algorithm is to take a random sample, and using a nearest neighbor search, to identify an identical class variable, which is defined as a NEAREST HIT (H), and also a different class variable that is nearest to the sample, defined as a NEAREST MISS (M) [93]. The iterative process of Relief estimates attribute weights $W[A_i]$ based on their similarity to a given class, where A_i, represents a unique attribute within the data set. The general form of the algorithm can be represented as follows [93].

Given m - desired number of sampled instances, and p - number of attributes,

1. set all weights $W[A_i]$:= 0.0;

2. for j := 1 to m do begin;

3. randomly select an instance X;

4. find nearest hit H and nearest miss M;

5. for i := 1 to k do begin;

6. $W[A_i]$:= $W[A_i]$- diff(A_i;X;H)/m + diff(A_i;X;M)/m;

7. end;

8. end;

Where the *diff* function above measures the difference between the attribute being evaluated A_i taken from the randomly selected instance X, and the value of that same attribute given the closest hit (H) or closest miss (M). For discrete attributes, if the value of the attribute (A_i) of the randomly selected instance (X) matches that of the nearest hit (H) or nearest miss (M), then the *diff* value is 0 (meaning values are identical), otherwise 1 meaning they are different. For continuous attributes, the actual difference is used and then normalized on a scale of [0,1].

By its design, the ReliefF attribute weighting technique does not constrain the attributes to non-negative values. Therefore, the weights will first be normalized based on the mini-max normalization [95]. For a given vector of attribute weights [w_1, w_2,..., w_p], the weights are normalized using the following formulation:
For weight i=1,. . . ,p

$$w_i' = \frac{w_i - min_w}{max_w - min_w}(new\_max_w - new\_min_w) + new\_min_w \tag{3.1}$$

Here,

w_i': The newly transformed weight (i) of attribute (i)

min_w: The minimum value in the vector of ReliefF attribute weights

max_w: The maximum value in the vector of ReliefF attribute weights

new_max_w: The maximum value of the new range

new_min_w: The minimum value of the new range

The new vector of weights **w'** determines the level of importance for each target vector ($\mathbf{T}^{C_j}$) at the engineering design level.

The data set with the newly updated weighted attributes will be used to:

1. Weight clusters generated by the X-means clustering approach (discussed in the following section).

2. Serve as attribute target weights for the engineering product design model.

X- Means Clustering

The X-means clustering algorithm in data mining is an enhancement of the k-means clustering algorithm [96]. Before investigating the X-means clustering algorithm and its significance in product family optimization, the k-means algorithm will be briefly described.

Given a raw data set of unique customer preferences (operating conditions), the k-means algorithm attempts to partition the original data set into k subsets of the data, where k represents the number of unique subsets or in the appropriate data mining terminology, clusters [97, 98]. Each cluster contains a centroid, with data points of the cluster associated with this centroid. It is important to note that the number of clusters in the k-means algorithm is given *a priori* as a user defined input. In the context of product family design, this would be analogous to the engineering design team specifying the number of product platforms that the customer's product operating requirements must adhere to. Rather than design teams making postulations about the raw data set, a more natural process would be for the inherent patterns of the raw data set to help guide the product platform number (this is one of the contributions of the X-means data mining technique). Although there have been many enhancements to the k-means since its conception [99, 100, 101], the basic underlying mathematical formulation can be represented as follows:

$$f = \sum_{j=1}^{K} \sum_{x_i \in S_j} ||x_i - c_j||^2 \tag{3.2}$$

Here,

1. S_j is a cluster of data points. Here, S will be defined as all instances in the raw data set and S_j would simply be a subset of this.
2. c_j is the centroid of a cluster S_j.
3. x_i is a data point existing within a cluster.
4. K is the total number of clusters (specified *a priori* by the user).

The iterative process of the k-means algorithm begins by initially selecting the desired number of clusters (S_j) and making an initial guess of the cluster centroid values (c_j) [97]. The next stage involves assigning a data point to the closest cluster centroid and centroid value (if necessary) by minimizing the error function in Equation (3.2) until negligible deviation occurs with each iteration.

The X-means clustering algorithm aims to improve on three key areas of the k-means algorithm [96].

1. Eliminating the need for number of clusters to be known *a priori*
2. Improving the computational scalability

3. Enhancing the search criteria for updating cluster centroids

The process by which X-means achieves these improvements is in part based on its selection criterion to determine when to add or replace a specific cluster centroid with child centroids. Child centroids originate from splitting the original solution of a k-means iteration and determining if the child clusters more accurately represent the data points once belonging to the parent centroid [96]. The posterior probabilities will be used to rank the models $\Pr[M_j|D]$, where D represents the given data set and M_j represents each model with a given cluster size k**.** The Bayesian Information Criterion (BIC) is used by X-means to rank which model is a more accurate representation of the original raw data set. Mathematically, the Bayesian Information Criterion (BIC) is represented as follows [102, 96]:

$$BIC(M_j) = -l_j(D) + \frac{p_j}{2}\log R \tag{3.3}$$

Here,

1. $l_j(D)$: is the log likelihood of the data taken at the maximum likelihood point.
2. D: represents the given data set.
3. p_j: represents the number of parameters in M_j.
4. R: is the total number of data points of candidate centroids.

Relevance of X-means to Engineering Product Architecture Design

Engineering design problems involving a wide range of operating states specified by customers can benefit from X-means clustering by identifying appropriate product functionality criterion for developing a product architecture and subsequent product family. The X-means clustering technique eliminates the need to guess the number of product architectures needed for a particular customer pool by analytically generating the appropriate number of clusters (product architectures) with corresponding product functionality specifications. A user instead specifies a broad range for the number of clusters and X-means will identify the optimal cluster, given the natural patterns within the data set [96]. This will ensure that the resulting product family will be a true representation of the data set for which the designs are made. Figure 3.2 illustrates how the cluster centroids of the X-means data mining clustering approach are integrated into the engineering design. The product centroids illustrated in Figure 3.2 represent the individual vectors of attribute value solutions that best describe *similar* groups of customers within the raw data. Each unique product centroid will form a vector of product preference targets used to guide the product architecture optimization process. The engineering design illustrated in Figure 3.2 represents the design of individual products based on the

X-means cluster centroids where each product will have unique functionality characteristics that aim to satisfy the overall customer preference targets. Section 3.1.2 of this work describes how product variants are then designed based on underlying product architecture under the notion of component sharing.

3.1.2 Engineering Design Optimization of Product Family

Step 1: Individual Product Design Optimization

The results from the data mining stage provide product design engineers with several vital pieces of information. First, the results from the X-means clustering represent the vector of product attributes that form the product design targets ($\mathbf{T}^{C_j}$) around which a product architecture is designed (Step 1 in Figure 3.1). Product design targets can range anywhere from physical product dimension targets such as length or width to product performance targets such as efficiency or speed.

The second vital piece of information from the data mining stage is the relevance of each attribute target to the customer as determined by the ReliefF attribute weighting technique. That is, for each attribute target vector ($\mathbf{T}^{C_j}$), there will be an accompanying vector of attribute target weights **w'**. The engineering product architecture optimization is comprised of the detailed engineering design model and incorporates the results from the data mining stage that help guide the product architecture design. Here, local design variables are used to model the physical dimensions and performance objectives of the product architecture subject to engineering design constraints.

The general mathematical model for the engineering product architecture optimization is as follows:

Note: The deviation is measured by the squared L-2 norm, which will be used throughout the engineering optimization models presented in this work. For example: $\|x-y\|_2^2 = \sum_i (x_i - y_i)^2$

For the *k*th product architecture,

Minimize

$$F(x)_{Architecture_{(k)}} = f_k + \mathbf{w}' \left\| T^{C_j} - R_k^{Eng} \right\|_2^2 \tag{3.4}$$

Subject to:

$\mathbf{g}_k(\mathbf{x}_k) \leq \mathbf{0}$

$\mathbf{h}_k(\mathbf{x}_k) = \mathbf{0}$

Here,

f_k Local product design objective function (s), a function of local design variables: $f_k(\mathbf{x}_k)$.

$\mathbf{T}^{C_j}$: Vector of product attributes represented by the cluster centroid in the data mining model. That is, for cluster centroid C_j=[A_1, A_2,..., A_p] target is set as $\mathbf{T}^{C_j}$ where A_1, A_2,..., A_p represent attribute values for a given centroid C_j.

w' : The vector of newly transformed ReliefF weights for target vector $\mathbf{T}^{C_j}$.

$\mathbf{R}_k^{Eng}$: Vector of engineering responses based on the formulation of the engineering design model. $\mathbf{R}_k^{Eng}$ is a function of local design variables $\mathbf{x}_k$, and is represented by $\mathbf{R}_k^{Eng}$ $(\mathbf{x}_k)$.

$\mathbf{g}_k$: Inequality design constraints bounding the product architecture model.

$\mathbf{h}_k$: Equality design constraints bounding the product architecture model.

k: The kth candidate product architecture determined by the results of the X-means clustering.

K: The total number of cluster centroids C_j that exist for the X-means clustering solution.

Note: It is important to note that although there may be K candidate product architectures to investigate, there may not always be a feasible design solution for the *kth* product architecture as generated product preference requirements may be too demanding, given the constraints of the engineering design model. That is, at optimality $k = K$.

Step 2: Component Sharing Through X-Means Clustering

If a feasible design solution exists after Step 1, X-means data mining clustering technique is once again employed, this time to determine the most *similar* product architecture design solutions within the product portfolio. While the first X-means clustering technique helped identify the similar groups of attributes in the raw data, the X-means clustering employed in Step 2 will help identify the similar groups of design variable values among the feasible product architecture design solutions (Step 2 in Figure 3.1).

For a given vector of design variables $(\mathbf{x}_k)$ of an optimal product architecture solution, (where the objective function $F(\mathbf{x}_k)$ of product architecture (k) has been minimized given the external targets $\mathbf{T}^{C_j}$ and the local objective(s) $f_k(\mathbf{x}_k)$), the goal is to determine the similarity among product architecture variable solutions. The notion is that the closer the optimal design solutions are, for example [$(\mathbf{x}_k)$ and $(\mathbf{x}_{k+1})$], the more likely these product architectures may be able to share certain design components.

Step 3: Product Family Optimization with Shared Design Components

The third and final step in the proposed product family design methodology aims to reduce the product portfolio cost by sharing certain components among product architectures, thereby creating a family of products (Step 3 in Figure 3.1). Since the component sharing decision is inherently a combinatorial problem, Step 2 of the design methodology eliminates the need to search all possible component sharing combinations by guiding the component sharing decisions based on the optimal solution of each resulting product architecture. Once similar product architectures have been identified by the X-means technique in Step 2, the component variables are identified and modeled as *linking variables* $(\mathbf{y}_{s,k})$ in the quasi-separable bi-level problem. The model in Step 1 is adapted into a bi-level hierarchical optimization model where *level 1* strictly handles the coordination of the linking variables and *level 2* still remains the product

architecture design level, but this time including the linking variable targets as part of the objective function. The bi-level design problem is modeled based on the quasi-separable problem [103, 104, 105, 106]. A bi-level model is presented which comprises the component sharing coordination model at the upper level and the individual product design model at the lower level. At the component sharing level, updated linking variable values are distributed among product variants in an iterative manner until a feasible solution is achieved that is common among all product variants. If a feasible design solution does not exist for a given sharing scenario (that is, linking variable value $\mathbf{y}_S$ does not converge to a solution shared by all products), the original product design solutions (without shared variables) from Step 1 are kept.

Upper Level: Component Sharing Coordination

The *Upper Level* (Component Sharing Coordination) handles the coordination of linking variables to each of the product variants. In an effort to minimize design costs, certain design intensive and costly product components are shared among the different product variants. Under the quasi-separable formulation, these are represented as the linking variables ($\mathbf{y}_{s,k}$). The sharing strategy is handled at the component sharing level wherein updated linking variable values from the lower level product architecture design are cascaded up to the component sharing level. Constraint Equation (3.5) is formulated as an inequality rather than an equality constraint due to numerical difficulties reported in the literature of equality constraint based bi-level formulations that fail to satisfy the standard Karush-Kuhn-Tucker (KKT) conditions for a constrained optimization problem [107].

Minimize ε_y

Subject to:

$$g1 : \sum_{k \in Q} \left\| \mathbf{y}_s - \mathbf{y}_{s,k}^{Eng} \right\|_2^2 - \varepsilon_y \leq 0 \tag{3.5}$$

Here,

$\mathbf{y}_s$: Linking variable at the *Upper Level*. In essence, $\mathbf{y}_s$ is simply a coordination variable ensuring that at the optimal solution, all of the subsystems attain the same value. Equation (3.5) is always active in the above formulation so solving for $\mathbf{y}_s$, it can be observed that at each iteration $\mathbf{y}_s$ assumes the average value of the linking variable(s) being shared across the products within the product family.

$\mathbf{y}_{s,k}^{Eng}$: Linking variable value at the *Lower Level* cascaded to the *Upper Level*. This is constant at each iteration in the above formulation that is subsequently updated at the engineering product architecture optimization level after each iteration.

k: The kth candidate product architecture that has been identified for component sharing.

Q: The total number of products that exist in a particular *candidate product family*. This is based on the X-means

cluster solutions described in Step 2. The term *candidate product family* is used because until a feasible design solution can be achieved for the shared component case, these Q products will remain unique products within the product portfolio (Note that $Q = K$ which simply means that the number of candidate product families cannot exceed the total number of unique products that initially exist).

ε_y: Deviation tolerance between linking variables. For each shared variable, another constraint g(i) is added based on a similar formulation as equation (3.5) and add another tolerance variable in the objective function to represent this additional shared variable.

Lower Level: Product Family Optimization

In the kth subproblem,

Minimize

$$F(x)_{Architecture_{(k)}} = f_k + \mathbf{w}' \left\| T^{C_j} - R_k^{Eng} \right\|_2^2 + \left\| y_s^U - y_{s,k} \right\|_2^2 \tag{3.6}$$

Subject to:

$\mathbf{g}_k(\mathbf{x}_k, \mathbf{y}_{s,k}) \leq \mathbf{0}$

$\mathbf{h}_k(\mathbf{x}_k, \mathbf{y}_{s,k}) = \mathbf{0}$

Here,

f_k: Local product design objective function (s).

$\mathbf{T}^{C_j}$: Vector of product attributes represented by the cluster centroid in the data mining model. That is, for cluster centroid C_j=[A_1, A_2,..., A_p] target is set as $\mathbf{T}^{C_j}$ where A_1, A_2,..., A_p represent attribute values for a given centroid C_j.

w' : The vector of newly transformed ReliefF weights for target vector $\mathbf{T}^{C_j}$.

$\mathbf{R}_k^{Eng}$: Vector of engineering responses based on the formulation of the engineering design model. $\mathbf{R}_k^{Eng}$ is a function of local design variables $\mathbf{x}_k$, and is represented by $\mathbf{R}_k^{Eng}$ ($\mathbf{x}_k$).

$\mathbf{g}_k$: Inequality design constraints bounding the product architecture model.

$\mathbf{h}_k$: Equality design constraints bounding the product architecture model.

k: The kth candidate product architecture determined by the results of the X-means clustering.

K: The total number of cluster centroids C_j that exist for the X-means clustering solution.

$\mathbf{y}_S^U$: Linking variable target value cascaded down to the *Lower Level* from the *Upper Level*; a constant value at each iteration that is subsequently updated with each successful iteration.

$\mathbf{y}_{s.k}$: Linking variable at the *Lower Level*. This is local to the kth model and attempts to match the value of $\mathbf{y}_S^U$ at each iteration.

The overall flow of the proposed product family optimization is succinctly described below:

Bi-level Product Family Optimization

Step 1:

Given **w'** vector of weights and $\mathbf{T}^{C_j}$ targets, where length(**w'**)=length($\mathbf{T}^{C_j}$) and K cluster centroids:

1. Solve K engineering design problems (**with no** linking variables $\mathbf{y}_{s,k}$) weighting each $||T^{Cj} - R_k^{Eng}||_2^2$ based on ReliefF;

2. If solution exists for the Individual Product Design Optimization Model

(i.e., *optimal* $||T^{C_j} - R_k^{Eng}||_2^2$ solution while satisfying local objectives and constraints);

3. Optimal solution found for weights **w'** and targets $\mathbf{T}^{C_j}$ without sharing components;

Step 2:

4. Employ **X-means** clustering to identify candidate product families based on solution similarities from **Step 1**;

Step 3:

5. Solve bi-level quasi-separable problem (component sharing among products) using the *Upper Level-Lower Level* formulation **with** linking variables $\mathbf{y}_{s,k}$;

6. If feasible solution exists (i.e., *optimal* $||T^{C_j} - R_k^{Eng}||_2^2$and $||\ y_s - y_{s,k}^L\ ||_2^2$ at the *Lower Level* and also *optimal* ε_y at the *Upper Level,* (ε_y should be close to 0 at the *Upper Level*, indicating a feasible shared component among product variants within a product family)) ;

7. Optimal solution found for weights **w'** and targets $\mathbf{T}^{C_j}$ and linking variables $\mathbf{y}_{s,k}$ for each product variant;

8 Else, solution does not exist for linking variable scenario; that is, sharing $\mathbf{y}_{s,k}$ is not feasible for product variants, therefore keep initial solutions found from **Step 1**;

9. end;

10. end;

3.2 Application: Aerodynamic Particle Separator Case Study

Indoor Air Quality (IAQ) is becoming an increasing concern for human health. Particulate matter is a leading cause of human respiratory illness in addition to degrading the performance of Heating Ventilation and Air Conditioning (HVAC) systems (W.H.O.). As a result, abatement technologies for these aerosols are in high demand. Aerodynamic particle separators are filter-less air cleaning devices that can be capable of removing micron size particles with low energy consumption and minimal maintenance [108]. Determining the optimal aerodynamic particle separator design for a specific application is challenging when taking into account its unique system requirements and environmental conditions.

Table 3.1: Design variable notation for aerodynamic particle separator

Variable	Units	Description
r_1	meters (m)	Inner tube radius
r_2	meters (m)	Outer tube radius
L_S	meters (m)	Maximum pressure drop
α	Radians (rad)	Vane discharge angle
L_C	meters (m)	Length of converging gap
r_3	meters (m)	Radius of exit tube
L_E	meters (m)	Length of exit tube
N	#	Number of units in parallel

3.2.1 The Engineering Design Problem

Aerodynamic Particle Separator Design

To demonstrate the effectiveness of the proposed design methodology, the design of a uniflow type particle separator is investigated (illustrated in Figure 3.3). The basic design of this device can be partitioned into three sections: (1) Vane section, (2) Straight Region and (3) Converging Region/Dust Bunker. These sections are defined by eight design variables as shown in Figure 3.3 and Table 3.1).

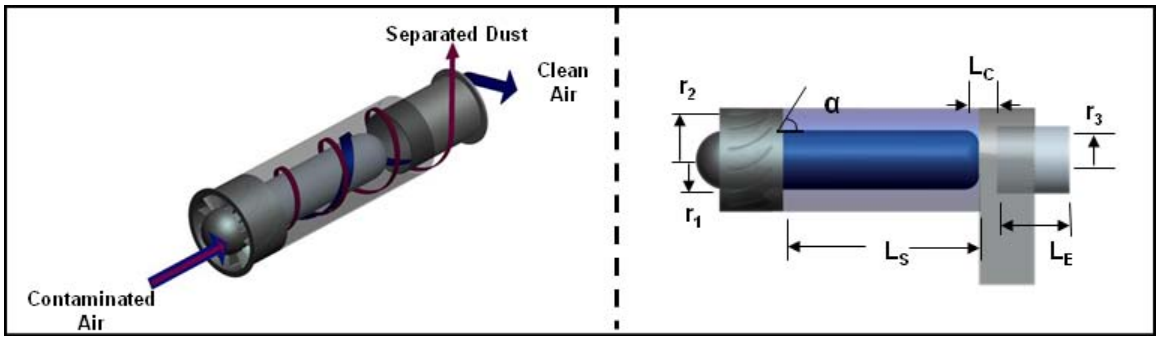


Figure 3.3: Uniflow type aerodynamic particle separator flow pattern and design variables

The performance of an aerodynamic particle separator design is strongly dependent on system requirements and environmental conditions. System requirements such as the air cleaning efficiency, pressure drop (thus power consumption), air flow rate and overall device size contribute to the design objectives and directly define the constraints for a given application. Environmental conditions, including the air properties and contaminant particle size distribution can have a significant impact on the performance of a particular design and must also be incorporated into the system model [109]. Together, these two groups can be used to characterize a given application or operating state. These factors can vary by an order of magnitude between different applications, thereby complicating the design pro-

Table 3.2: Snapshot of aerodynamic particle separator data set consisting of 1000 states

	System requirements					Environmental conditions					
	Q (m3/s)	Δpmax (Pa)	L_{max} (m)	$A_{F\,max}$ (m^2)	N_{max} # units	$F(d_p)$ (%)	ρ_p (kg/m^3)	T_{air} (°C)	P_{air} (kPa)	Efficiency %	Price $
State 1	1.58	250	1	0.5	50	A1	2650	20	101	82%	$1,200
State 2	1.48	200	0.3	0.1	3	A4	2650	0	99	85%	$450
State 3	1.27	1500	1.5	1.5	16	Limestone	2700	500	200	90%	$900

cess. The product architecture design objective will, therefore, be to minimize cost while satisfying external product preference targets and local design constraints. The broader enterprise portfolio objective will be to minimize overall product design and development costs by capturing the component sharing opportunities that exists within the product portfolio.

3.2.2 Data Mining Product Preferences

Raw Data Set of Product Operating States

A data set of 1,000 operating states was generated to simulate the large variation in physical requirements and environmental conditions characterizing the broad range of applications in which aerodynamic particle separators are frequently employed [109]. Table 3.2 represents a snapshot of the 1,000 operating states with distinct product attributes and environmental conditions represented by each column. Section 3.3 of this work presents the results from both the ReliefF attribute ranking algorithm and the X-means data mining clustering approach and demonstrates how the data mining process influences product family design efforts.

ReliefF Attribute Weighting

The results from the ReliefF attribute ranking approach in Table 3.3 reveal that the two engineering design targets – efficiency (ζ) and flow area (AF_{max}) – have normalized weights of 0.1687 and 0.0785 respectively. The other attributes in Table 3.1 are used as design parameters in the engineering design model and also play a significant role in the overall optimal solution. The efficiency (ζ) and flow area (AF_{max}) are selected based on the type of engineering problem being solved (in other applications, one may choose to set all attributes in the data set as targets for the product architecture design model). This vital information is a data preprocessing step that will help generate product cluster centroids that take into account the weighted attribute preferences of the different operating states given by the raw data set.

Note: ReliefF results were obtained using Weka version 3.5.8 [2] and it took approximately 20 seconds running on a Intel Pentium Duo 2.5 GHz Processor. The normalized attribute weights are based on Equation (3.1) in Section 3.1.1.

Table 3.3: Attribute ranking of raw data set via ReliefF algorithm

Attribute Rank	Attribute Name	Attribute Weight	Normalized Attribute Weight
Highest	Nmax	0.0881	1.0000
↓	Efficiency	0.0110	0.1687
↓	Tair	0.0100	0.1575
↓	Pair	0.0099	0.1570
↓	Q	0.0073	0.1287
↓	Dpmax	0.0036	0.0888
↓	Afmax	0.0026	0.0785
↓	Lmax	0.0018	0.0695
↓	Rhop	-0.0006	0.0433
↓	Operating States	-0.0031	0.0169
Lowest	Fdp	-0.0046	0.0000

Data Mining X-means Clustering Results

The X-means clustering results reveal that a total of five clusters most accurately represents the similarities in the data set of 1,000 operating states. The results from Table 3.1 represent the product design targets and parameters for the product portfolio of aerodynamic particle separators. Initially, each product centroid will be used to design an individual aerodynamic particle separator. Component sharing benefits will then be presented based on the vane section component. [Results attained using Weka version 3.5.8 [2] and Data to Knowledge D2K [1]].

3.2.3 Engineering Design Optimization of Product Family

Step 1: Individual Product Design Optimization

The X-means clustering algorithm generates k=1,..., 5 clusters (Table 3.4, each with unique centroids C_j). Based on the results from the X-means clustering, and the ReliefF attribute weights accompanying each cluster centroid, engineers can now determine whether an optimal product design solution exists based on the aerodynamic particle separator response model.

The aerodynamic particle separator objective function attempts to match the particle separation efficiency target ($\zeta_k^{C_j}$) and the flow area target (AF_{C_j}) generated from the X-means clustering results while at the same time minimizing product design and manufacturing cost objective. The attributes within a cluster centroid (C_j) will form the design/environmental parameters of the model. The efficiency model selected was initially developed by Zhang [108] and later augmented by Barker [109]. In this model, the flow is assumed to be fully turbulent and the steady state particle motion results from a balance between the centrifugal force and aerodynamic drag in the Stokes regime [109]. The vector **x** contains the eight design variables as described by Table 3.1 and Figure 3.3. The cost function was based

Table 3.4: Product cluster centroids based on X-means clustering algorithm

	System requirements					Environmental conditions						
States	Q (m3/s)	Δpmax (Pa)	L_{max} (m)	A_{Fmax} (m^2)	N_{max} # units	$F(d_p)$ (%)	ρ_p (kg/m^3)	T_{air} (°C)	P_{air} (kPa)	Efficiency %	Price $	
285	1.20	463	0.71	0.60	36	Limestone	2226	25	100	78%	$962	Product Centroid 1
765	3.39	1507	0.79	0.68	39	A4	2211	71	111	86%	$1,168	Product Centroid 2
750	3.36	1623	0.67	0.55	14	Limestone	2389	72	107	85%	$411	Product Centroid 3
456	1.94	911	0.72	0.62	24	A1	2278	45	102	80%	$632	Product Centroid 4
260	1.29	458	0.69	0.75	11	Limestone	2225	24	100	78%	$290	Product Centroid 5

on the estimated mass of material required and injection molding costs of the vane section. The material selected is an engineered polymer with a density of 1200 kg/m3 at a cost of $3.00 per kilogram. The injection molding cost is estimated at a fixed cost of $10,000 per design for the required capital equipment and labor. The efficiency model as a function of variables in $\mathbf{x}$ and particle size d_{p_i} is shown in Equation (3.7). The total efficiency for a given particle size distribution is then calculated by Equation (3.8).

$$\xi(x,d_{p_i}) = 1 - \exp\left(-\frac{\rho_p d_{p_i}^2 C_c Q \tan(\alpha) L_S}{9\eta(r_2^2 - r_1^2)}\right) \cdot \exp\left(\frac{\rho_p d_{p_i}^2 C_c \left(V_t^2 G_t(x) + V_z^2 G_r(x)\right)}{\eta V_z}\right) \tag{3.7}$$

$$\xi_T = \sum_{i=1}^{N} \xi(x,d_{p_i}) \cdot F(d_{p_i}) \tag{3.8}$$

Here,

Cc: Cunningham slip correction factor.

d_{p_i}: Diameter of particle (i), m.

$F(dp)$: Particle size distribution.

$G_t(x)$: Efficiency model geometric relationship between design variables, tangential acceleration.

$G_r(x)$: Efficiency model geometric relationship between design variables, radial acceleration.

ρ_p : Particle density, kg/m^3.

η : Air viscosity, Pas or kgm/s.

Q : Air flow rate, m^3/s.

V_t: Tangential velocity of particle mixture.

V_z: Axial velocity of particle mixture.

r_1: Inner tube radius.

r_2: Outer tube radius.

α: Vane discharge angle.

L_S: Maximum pressure drop.

The engineering design model for the aerodynamic particle separator can be mathematically represented as:

kth Aerodynamic Particle Separator Minimize:

$$F(x)_{Architecture_{(k)}} = w'_{\zeta} \left\| \zeta_k^{Cj} - \zeta_k^{Eng} \right\|_2^2 + w'_{AF} \left\| AF_k^{Cj} - AF_k^{Eng} \right\|_2^2 + Cost_k \tag{3.9}$$

Subject to:

Pressure drop constraint (g_1):

$$P_T(x) - P_{\max} \leq 0 \tag{3.10}$$

Face Area constraint (g_2):

$$4r_2^2 N - AF_{\max} \leq 0 \tag{3.11}$$

Product Length constraint (g3):

$$L_V + L_S + L_C + L_E - L_{\max} \leq 0 \tag{3.12}$$

Here,

AF_k : Maximum allowable face area perpendicular to air flow direction.

$w\zeta$': Efficiency ReliefF attribute weight.

w'_{AF}: Flow area (AF) ReliefF attribute weight.

L_{max}: Total allowable length of the system.

L_V: Length of vane section.

L_S: Length of straight region.

L_C: Length of converging region.

L_E*:* Length of exit tube.

PT(**x**): Total pressure drop of the system as a function of design variables **x**.

N: Number of aerodynamic particle separator units in one module.

P_{max}: Maximum allowable pressure drop (air flow restriction).

$Cost_k$: Total product cost represented as the summation of individual component costs.

Note: The design model is also bounded by a set of linear inequality constraints Ax = b and constraints Equations (3.10)-(3.12) that can be further expanded. A more detailed design model can be found in [109].

Step 2: Component Sharing Through X-Means Clustering

If an optimal solution exists for the aerodynamic product portfolio based on the X-means clustering targets, the next step is to determine whether additional costs savings can be realized by sharing the most design intensive components among different product architectures. The X-means clustering technique is employed to determine the similarities among the unique aerodynamic particle separator designs based on the solution results after **Step 1**. A successful sharing solution among products represents a unique product family. The results from the unique aerodynamic particle separator solutions can be seen in Table 3.5.

Step 3: Product Family Optimization with Shared Design Components

For the aerodynamic particle separator case study, the vane section is the most design intensive and costly component. The complex curved vanes must be injection molded, which requires a unique mold to be machined for each vane section design. By employing the X-means clustering technique, product engineers will be able to: (1) determine which product architecture designs are similar based on the solutions attained during Step 1 and (2) determine the number of candidate product families to include in the enterprise product portfolio based on the number of X-Means cluster centroids generated. The L2 norm distance measure used by X-means will favor those design solutions that are numerically close to one another. This will help guide the sharing decision of the vane section as products with close numerical values for the variables that define the vane section (vane angle α, the inner and outer tube radii r_1 and r_2) will be favored within a given cluster centroid.

Upper Level: Component Sharing Coordination The *Upper Level* (Component Sharing Coordination) of the Aerodynamic Particle Separator model will handle the coordination of the shared vane section among product families. The Component Sharing objective function will minimize the tolerance deviation variable of each shared variable. There are three variables that define the vane section, including the vane angle α, the inner and outer tube radii r_1 and r_2.

Minimize $\varepsilon_\alpha + \varepsilon_{r_1} + \varepsilon_{r_2}$

Subject to:

$$g1 : \left\| \alpha_s - \alpha_{s,k}^{Eng} \right\|_2^2 - \varepsilon_\alpha \leq 0 \tag{3.13}$$

$$g2 : \left\| r_{1,s} - r_{1,s,k}^{Eng} \right\|_2^2 - \varepsilon_{r_1} \leq 0 \tag{3.14}$$

$$g3 : \left\| r_{2s} - r_{2s,k}^{Eng} \right\|_2^2 - \varepsilon_{r_2} \leq 0 \tag{3.15}$$

Here,

α_s : Vane angle linking variable at the component sharing level.

$\alpha_{s,k}^{Eng}$: Value of vane angle linking variable response of Engineering Design Model for product k.

$r_{1s,k}$: Inner tube radius (r_1) linking variable at the component sharing level.

$r_{1_{s,k}}^{Eng}$: Value of Inner tube radius (r_1) linking variable response of Engineering Design Model for product k.

$r_{2s,k}$: Outer tube radius (r_2) linking variable at the component sharing level.

$r_{2_{s,k}}^{Eng}$: Value of Outer tube radius (r_2) linking variable response of Engineering Design Model for product k.

ε_α: Deviation tolerance variable between vane angle linking variable that is minimized in the objective function.

ε_{r1}: Deviation tolerance variable between inner radius linking variable that is minimized in the objective function.

ε_{r2}: Deviation tolerance variable between outer radius linking variable that is minimized in the objective function.

Lower Level: Product Family Optimization To minimize overall product portfolio costs, the number of unique vane section designs will be minimized by sharing this component with products that can attain a feasible design solution given this added objective. Equation (3.9) is reformulated to reflect the candidate product families and also the shared vane components among each of these products within a given product family (represented as linking variables).

Minimize:

$$\begin{aligned} F(x)_{Architecture_{(k)}} = w'_\zeta \|\zeta_k^{Cj} - \zeta_k^{Eng}\|_2^2 + w'_{AF} \|AF_k^{Cj} - AF_k^{Eng}\|_2^2 + Cost \\ + \|\alpha_{s,i} - \alpha_{s,k}^{Link}\|_2^2 + \|r1_{s,i} - r1_{s,k}^{Link}\|_2^2 + \|r2_{s,i} - r2_{s,k}^{Link}\|_2^2 \end{aligned} \tag{3.16}$$

Subject to: Constraints as defined in Equations (3.10)**,** (3.11) and (3.12)

3.3 Results and Discussion

3.3.1 Aerodynamic Particle Separator Optimization Results

Given the product design targets from the Data Mining X-means clustering step, the aerodynamic particle separator model first attempts to identify feasible design solutions for the efficiency (ζ_{Cj}), flow area (AF_{C_j}) targets and given physical and environmental (T_{air}, P_{air}, etc.) parameters for each unique cluster centroid (C_j). The aerodynamic particle

separator solutions in Table 3.5 reveal that a total of five unique products can be designed for the initial five cluster centroids targets generated by the X-means clustering with a total product portfolio cost of $173,910.

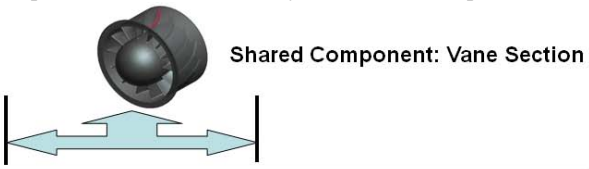
Table 3.5: Optimal solutions for individual aerodynamic particle separator designs*

Product	Design Variables									Product Efficiency	Product Unit Cost	# Units/ Cluster	Injection Mold Cost	Total Product Cost
	r_1 (meters)	r_2 (meters)	Ls (meters)	ά (rad)	Lc (meters)	r_3 (meters)	AF (meters)	L (meters)	N (# units)	ς (%)	$	(# units)	$	$
Particle Separator 1	0.1732	0.1936	0.1098	1.0400	0.0109	0.1163	0.1207	0.6000	4	77.9994	$69.01	226	$10,000.00	$25,595.41
Particle Separator 2	0.0707	0.0842	0.2709	1.0400	0.0088	0.0715	0.27971	0.6799	24	85.9915	$171.03	207	$10,000.00	$45,403.45
Particle Separator 3	0.0248	0.0991	0.3539	1.0400	0.0991	0.0527	0.45301	0.5499	14	84.9963	$86.53	173	$10,000.00	$24,970.02
Particle Separator 4	0.0691	0.0804	0.3696	1.0400	0.0804	0.0683	0.44998	0.6199	24	79.9955	$189.70	233	$10,000.00	$54,199.65
Particle Separator 5	0.1758	0.1936	0.0932	1.0400	0.0095	0.1068	0.10267	0.7500	5	77.9993	$85.36	161	$10,000.00	$23,742.41
Total Product Portfolio Cost														**$173,910.93**

If **Step** 1 of the product family design methodology is successful, engineers can further investigate the potential costs savings (**Steps 2 and 3)** that may be realized due to component sharing. The X-means clustering technique performed during **Step** 2 reveals that out of the five unique aerodynamic particle separator solutions, products **1 and 5** form a *feasible* unique product family cluster, products **2 and 3** another and finally product **4** cannot be shared with any other product and, therefore, reverts back to the original solution from **Step 1**. The cost of the injection mold manufacturing process presents an opportunity for the initial product portfolio of five unique products to be redesigned. The vane section of the product which is made through the injection molding process is shared among *similar* products existing in the original portfolio. In this case study, the decision to share the vane angle is known as a priori due to the high cost of designing each individual injection mold for the vane. **Step** 3 of the product family design methodology employs the X-means clustering algorithm to identify products that have similar vane design solutions. The decision to share the vane angle is an attempt to minimize the overall costs of the enterprise product portfolio by minimizing the number of unique vane sections needed for the five aerodynamic particle separators. Products successfully sharing a vane section will be considered a unique *product family* and each product existing in this product family, is defined as a *variant.* However, it must be noted that the cost savings benefits of component sharing using the product family approach to design may be offset by the decrease in the performance capabilities attainable by the newly designed product variants. This trade-off scenario will, therefore, be based on how much cost savings can be realized through component sharing and how much performance deviation can be accommodated by the customer.

The results in Table 3.6 reveal that sharing products 1 and 5, 2 and 3 (with product 4 being a separate unique design), reduces the total product portfolio cost to $163,150: a total savings of approximately $10,760 for this product portfolio design scenario. However, the efficiency of product 2 decreases from 85.99% with the individual optimization model solution (Table 3.5) to 85.84% with the component sharing product family model solution (Table 3.6). The level

Table 3.6: Optimal Solutions for Aerodynamic Particle Separator Product Families sharing the Vane Component*



Product	Product Design Variables									Product Efficiency	Product Unit Cost	# Units/ Cluster	Injection Mold Cost	Total Product Cost
	r_1 (meters)	r_2 (meters)	Ls (meters)	$\acute{\alpha}$ (rad)	Lc (meters)	r_3 (meters)	AF (meters)	L (meters)	N (# units)	ς (%)	$		$	$
Particle Separator Product Family 1														
Variant 1	0.1643	0.1935	0.2771	0.9532	0.0106	0.0884	0.5993	0.2877	4	77.9985	$81.54	226	$10,000.00	$28,428.21
Variant 5	0.1653	0.1935	0.2991	0.9533	0.0097	0.0812	0.7487	0.3088	5	77.9979	$105.10	161	Shared	$16,921.90
Particle Separator Product Family 2														
Variant 2	0.0908	0.1064	0.5559	0.9446	0.0071	0.0905	0.6797	0.5630	15	85.8444	$196.45	207	$10,000.00	$50,664.52
Variant 3	0.0903	0.1070	0.0509	0.9446	0.0072	0.0686	0.5495	0.0581	12	84.9991	$74.77	173	Shared	$12,935.74
Particle Separator Product Family 3														
Variant 4	0.069059	0.080357	0.369626	1.04	0.080357	0.068303	0.449983	0.619893	24	79.995506	$189.70	233	$10,000.00	$54,199.65
Total Product Portfolio Cost														**$163,150.02**

of allowable performance deviation will be dependent on customer expectations and the level of competition within the market space. Although a feasible design may not always exist for every sharing scenario (for example sharing a single vane component for each of the five products returned an infeasible solution), the benefits of investigating sharing strategies through the X-means clustering recommendations may prove beneficial as can be seen from the results in Table 3.6.

3.4 Conclusion

In this work, a comprehensive product family design methodology is presented that integrates realistic product operation data with the engineering design of complex products such as the aerodynamic particle separator. The data mining ReliefF algorithm is employed to determine the weights of each attribute. This information is then incorporated into the data mining X-means clustering algorithm in order to generate the number of clusters along with the cluster centroids that are inherent to the data itself. The results of the data mining clustering technique aid in determining the number of unique products to design for a group of highly diverse customers. With this clustering information, a product architecture can be designed that takes into account specific customer product functionality needs that are represented in a large data set. Further cost savings can be realized through a component sharing strategy that is achieved in this work by once again employing the X-means clustering technique to identify similar design solutions. The hope is to expand on the concepts presented in this work by enabling the feasibility of the product architecture optimization step to influence the generation of X-means cluster centroids. That is, local objective functions may be highly sensitive to certain local design variables which can be taken into account during the X-means clustering step.

Chapter 4

Decision Tree Classification for Product Concept Generation

4.1 Product Concept Generation Methodology

The proposed Data Mining Decision Tree Classification approach to product portfolio design takes large data sets of customer preference data and extracts meaningful product attribute information to help guide next generation product design and development process. The overall objective of maximizing company profit is realized when a feasible set of product variants is presented in the final solution process. The reduction of resources in this fast and highly efficient narrowing of product concepts will be demonstrated through a cell phone example where an entire product concept generation space of 576 (exhaustive combination of product attributes) product concepts is narrowed to only 46 through a decision tree data mining approach. The generated product designs are then subsequently tested for engineering feasibility. This is formulated as a multilevel optimization problem, where the generated predictive product concepts are first translated into functional specifications and set as targets at the engineering level for design validation. A feasible product design is therefore defined as one in which all customer preferences are satisfied, without violating engineering design constraints.

The entire product portfolio generation process is divided into two phases. *Phase 1* is the customer knowledge discovery process which entails customer data acquisition and processing and data mining for feasible set generation. *Phase 2* involves the product concept validation through multilevel optimization and finishes with a product portfolio selection. Figure 4.1 shows the overall flow of this process (The general flow on the left and the detailed flow on the right of Figure 4.1) starting with customer data acquisition and ending with enterprise portfolio selection. The details of the methodology are presented as follows:

4.1.1 The Concept of Novel, Previously Unknown Customer Information

The term *product concept* that is defined in the dissertation relates to the notion of novel, previously unknown customer information that data mining is well known for [27, 28, 32]. To illustrate this concept, a simple test data set is presented in Table 4.1. The data set contains 6 customer attributes (columns 1-6) with 1 predictor variable (Class variable in

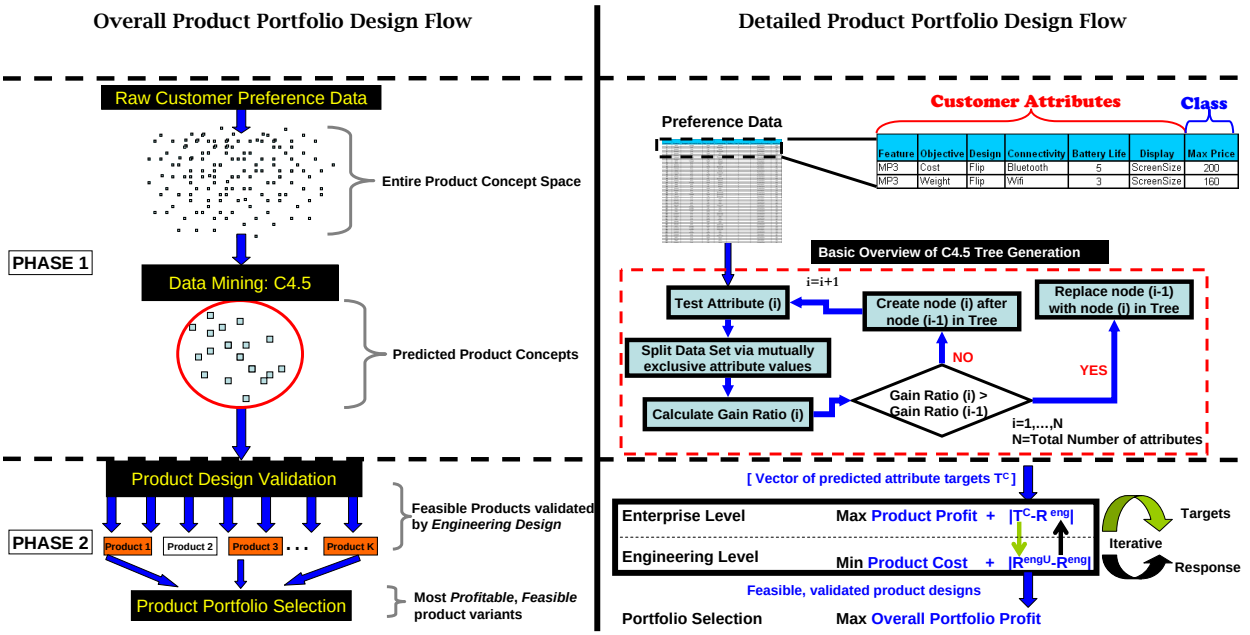


Figure 4.1: Overall flow of product portfolio optimization process.

column 7). Based on the attribute values in Table 4.1 of Feature, Objective, Design, Connectivity, Battery Life, and Display, there are a total of $3 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 2 = 216$ possible unique combinations (although only 10 out of the 216 exist in the sample data in Table 4.1). Two fundamental questions arise from observation:

- How can engineers determine novel attribute combinations without performing additional data acquisition procedures (customer surveys, focus groups, etc.)?
- How efficiently can engineers extract these new attribute combinations?

The term *novel* in this dissertation relates to information that is not readily observable or not explicitly defined within the data set but can be quantified through the proposed decision tree induction technique. The following product design question aims to illustrate how novel information can be extracted from a raw dataset.

- Given a specific attribute combination not existing within the data set: (For example referring to Table 4.1 in the dissertation, it can be observed that the combination of {Games, Weight, Flip, Bluetooth, 5 hours battery, ScreenSize} does not exist within the data set):

1. What price category (MaxPrice) would the above attribute combination fall under?
2. Are all of these attributes even needed in predicting the price category? That is, if only a subset of the attribute space is included {Games, Weight and Flip} instead of the entire attribute space {Games, Weight, Flip, Bluetooth, 5 hours battery, ScreenSize}, would it still result in the same price category (MaxPrice) prediction?

Table 4.1: Example data set of customer attributes

Feature	Objective	Design	Connectivity	Battery Life	Display	MaxPrice
MP3	Cost	Flip	Bluetooth	5	ScreenSize	200
MP3	Weight	Flip	Wifi	3	ScreenSize	160
MP3	Weight	Flip	Bluetooth	3	Resolution	160
MP3	Cost	Shell	Infrared	5	ScreenSize	80
Camera	Weight	Shell	Wifi	3	ScreenSize	120
Camera	Weight	Flip	Wifi	3	Resolution	120
Games	Cost	Flip	Bluetooth	5	Resolution	200
Games	Cost	Shell	Wifi	7	Resolution	200
Games	Weight	Flip	Infrared	5	ScreenSize	160
Games	Cost	Flip	Bluetooth	3	ScreenSize	160

The case study example in section 4.2.1 helps address these questions. For example the decision tree structure in Figure 3.2 reveals that for the *Games* phone, as long as the product also includes *bluetooth connectivity* and a *5 hour battery life*, a price of $120 would be predicted. Therefore if design engineers were aiming to design the next generation of *Games* phones to a customer market segment willing to pay $120, then these product attributes would make up the primary product architecture.

Another example of attribute knowledge discovery can be observed in Table 4.1. If a camera phone product were to be designed, rows 5 and 6 indicated that both camera phones, each with slightly different attribute combinations, yield a purchase price of $120. However, based on the C4.5 algorithm (explained in section 1.4.1), no additional attributes are needed to yield a camera phone price of $120 (see the initial partitioning in Table 4.2). Therefore from a product design perspective, no additional resources should be invested in improving additional design features that do not significantly influence the purchase decisions of a customer. This type of information is not readily observed in the raw data set and will enable design engineers to design the next generation of *Games* and *Camera* phones by including only the relevant attributes along with their predicted attribute levels. Such insights have the potential to save on manufacturing and materials costs as well as on the time and efficiency of the product design process.

4.1.2 Product Concept Generation using C4.5 Machine Learning Algorithm

The proposed product concept generation methodology employs Quinlan's C4.5 data mining tree generation algorithm for attribute characterization [110]. The algorithm is based on the *Divide and Conquer* [111, 110] technique that decomposes a set of training cases T with class variables $\{C_1, C_2, ..., C_N\}$ until the partitioning yields a collection of cases that predicts a single class variable C_i. Each subsequent decomposition of the tree tests a single attribute that has outcomes $\{O_1, ..., O_K\}$ that are mutually exclusive to one another [111]. When applied to product portfolio optimization, the class variable can be thought of as the overall performance criteria (determined by the enterprise decision maker) influencing product launch decisions. The class variable selected by the enterprise decision maker can range from a *Price metric* (later to be demonstrated in the cell phone example) to a *Weight* or *Dimensionality* metric, etc.

The manner in which attributes are selected during each stage of tree decomposition is the fundamental strength of the C4.5 algorithm and the primary reason why this data mining technique is so successful when applied to the product portfolio paradigm. The term attribute can be thought of as the quantifiable product requirements of a customer. Examples of attributes may be *minimum fuel economy expectations* (miles per gallon) in the context of automotive design or the *battery life* expectations of a hand held device. The tree termination criterion eliminates the need for an exhaustive search of all possible attribute combinations, and when applied to multi-level optimization formulation in product development, significantly improves on the time and efficiency of developing a portfolio of products. (Demonstrated later in the cell phone example).

C4.5 Gain Ratio Criteria:

To avoid an exhaustive search of all possible attribute combinations, a systematic approach is to partition the data and identify what attribute to split in the most efficient manner so as to gain the most information about the class variable. For a given training set T, a particular attribute that has K possible outcomes $\{O_1, ..., O_K\}$ [111] can be tested. If S is defined to be any set of cases (which can either be the entire training set T or a *subset* of T), then the occurrence of a particular class variable C_i can be denoted by

$$freq(C_i, S) \tag{4.1}$$

This is simply the number of times a particular class occurs in a given data set. The information gained by splitting a particular $attribute_i$ gets its foundation from classical information theory that states: *The information conveyed by a message depends on its probability and can be measured in bits as minus the logarithm to base 2 of that probability* [110] . If $freq(C_i, S)$ determines the number of occurrences of a particular class, then the probability of randomly selecting this class over the entire set of S cases would simply be:

$$\frac{freq(C_i, S)}{|S|} \tag{4.2}$$

where $|\ S\ |$ represents the total number of cases in the data set S.

Following the definition of *information conveyed*, the information that this particular example conveys can be represented as [111]

$$-\log_2(\frac{freq(C_i, S)}{|S|}) \quad [bits] \tag{4.3}$$

It is interesting to note that the range of the class variable C can be set by the enterprise decision maker depending

on the desired objective of the company. If customer willingness to pay is the performance metric (Class) to be predicted, then this can be partitioned into $\{C_1, C_2, ..., C_N\}$, that is if the data is obtained through a direct customer survey approach.

Later in the cell phone example, the primary criterion for selecting one device over the other is the maximum price a customer is willing to pay for that particular design: MaxPrice, as it is abbreviated in the example is therefore the class variable to be predicted. To measure the average amount of information needed to identify the class (for example, all values of MaxPrice ranging from [$40 $80 $120 $160 $200]) of a case in a training set, the classes are summed relative to their frequencies in the data set [110].

$$info(S) \quad = \quad -\sum_{i=1}^{N}(\frac{freq(C_i,S)}{|S|}) \cdot \log_2(\frac{freq(C_i,S)}{|S|})[bits] \tag{4.4}$$

Note: T represents the entire set of training cases while S represents any set of cases within T. Therefore, the above formula can be used to calculate the *information* of subsets of T or the entire data set T. *info(T)* therefore measures the average amount of information required to identify the class of a case in T by summing over the product of all the class probabilities and their information as defined by equation 4.4 [111]. To test the amount of information gain of a particular attribute, the attribute is partitioned into its respective mutually exclusive outcomes.

After partitioning T into k possible outcomes for a specified test X (attribute selection), the expected information requirement is the summation of all subsets as given by [112]:

$$info_x(T) \quad = \quad \sum_{j=1}^{k} \frac{|T_j|}{|T|} \cdot info(T_j) \tag{4.5}$$

The *gain* can therefore be defined as the difference in the total average information required to identify a class in the training set minus the information achieved by testing a particular attribute [113]:

$$gain(X) \quad = \quad info(T) - info_x(T) \tag{4.6}$$

The above equation itself is an optimization problem where the objective is to maximize the information gain, subject to the constraints of the algorithm sequence. Due to the fact that certain attributes may have significantly greater outcomes, this metric alone may not be sufficient as it may skew the predictive capabilities of the algorithm in favor of attributes with greater outcomes. A more accurate predictor of the information that is gained by partitioning T is the *gain ratio criterion* that is defined as [111]:

$$GainRatio(X) = \frac{gain(X)}{SplitInfo(X)} \tag{4.7}$$

Where:

$$SplitInfo(X) = -\sum_{j=1}^{n} \frac{|T_j|}{|T|} \cdot \log_2 \frac{|T_j|}{|T|} \tag{4.8}$$

The *gain ratio* represents the proportion of information (i.e., scaled information) generated by the split that is useful in predicting the class variable [111].

The partitioning of a problem into subproblems (i.e., generating concepts) will be terminated when there is only one class in that particular branch [111]. Pruning of subsequent branches can occur if replacing a branch with a leaf will reduce the % error of that node and ultimately the entire branch [114].

C4.5 Discretization of Continuous Attributes

The C4.5 algorithm performs discretization and tree induction concurrently and is therefore not a user defined input [115, 116]. For the case of a continuous attribute within a given data set (for example, a price or weight variable), a binary split is determined for each attribute based on *minimal entropy* criteria [117, 116]. More recent contributions to the C4.5 discretization of continuous attributes employ the Minimum Description Length (MDL) to help minimize the bias that may be inherent in the underlying *gain ratio* criterion explained above.

Since discretization of continuous attributes is handled during the C4.5 tree generation approach [110, 117] , the resulting attribute combination represents the most appropriate discretization to predict the class variable. Since C4.5 discretization is limited to the attribute space, and does not include the class variable, enterprise decision makers may opt to choose a discrete variable to serve as the class variable. In the cell phone example presented later in the work, the class variable represents pricing information gathered through an online interactive customer survey and therefore is discrete based on the design of the survey. On the other hand if the data set comprises of revealed preference data such as electronic store purchases or online transactions, the pricing information may be inherently continuous and can therefore either be discretized during the data mining preprocessing step explained in section (1.4.1) or serve as an attribute in the C4.5 formulation (another class variable such as "purchase phone: *Yes* or *No*," may serve as the class variable in this scenario). One also has the option to employ other data mining techniques that can handle continuous class variables such as M5 Prime [118] or Classification and Regression Trees (CART) [119] which could then be applied to other product design scenarios containing continuous class variables.

Interpretation and Evaluation

PHASE 1 of the product portfolio formulation process provides design engineers with three critical pieces of information vital to the product concept validation process (PHASE 2).

- **Set of candidate product concepts:** Represented as a unique combination of customer attributes.
- **Class variable prediction:** The predicted performance evaluation for product concept (j). In the example, this is denoted by **MaxPrice.**
- **Aggregated demand for a particular product concept:** Represented by the total supported cases for a particular predicted class variable (represented as a leaf in the C4.5 decision tree).

4.1.3 PHASE 2: Product Concept Validation through Multilevel Optimization

The product concepts generated by the C4.5 decision tree data mining technique in PHASE 1 need to be validated to ensure that such performance expectations can be realistically designed. This is modeled as a multilevel optimization problem by adopting the Analytical Target Cascading (ATC) [89] multilevel optimization approach (although the methodology is not limited to the ATC only). Phase 2 ends with a portfolio selection decision after *feasible* product concepts have been validated by the interactions between the enterprise level and engineering level.

Enterprise System Level

This is where the profit of each individual architecture is calculated. This level includes the set of generated product concepts that are directly incorporated into the engineering product design process. Also included in the Enterprise System level is the market demand information predicted for a particular product variant *(j)*. Mathematically, this is represented as:

Given

$$\mathbf{T}^C, MaxPrice_{\{j\}}, d_j, \mathbf{R}^{Eng^L}, Cost^L_{Architecture\{j\}}$$

$$\min \quad -\pi_{Architecture_j} + \|\mathbf{T}^C - \mathbf{R}^{Eng}\|_2^2 + \varepsilon_{\mathbf{R}} + \varepsilon_C \tag{4.9}$$

with respect to

$$\mathbf{R}^{Eng}, Cost_{Architecture\{j\}}$$

subject to

$$h1 : \pi_{Architecture_j}$$

$$-d_j \cdot \left(MaxPrice_{\{j\}} - Cost_{Architecture\{j\}}\right) = 0 \tag{4.10}$$

$$g1 : \|\mathbf{R}^{Eng} - \mathbf{R}^{Eng^L}\|_2^2 - \varepsilon_{\mathbf{R}} \leq 0 \tag{4.11}$$

$$g2 : \|Cost_{Architecture\{j\}} - Cost^L_{Architecture\{j\}}\|_2^2 - \varepsilon_C \leq 0 \tag{4.12}$$

Enterprise Level: Variable Notation Definitions

$\mathbf{T}^C$: Architecture targets (set of attribute combinations) predicted by C4.5 Decision Tree model.

d_j: Represents the customer demand for product concept j predicted by the C4.5 data mining tree generation. (Conceptually, this represents the number of cases supporting the final attribute partitioning, yielding a single leaf, i.e., class prediction).

$MaxPrice_j$: is the single class variable predicted by the continual partitioning of the set of training data until a single leaf (class) is achieved.

$\mathbf{R}^{Eng^L}$: Engineering performance response target from the engineering subsystem level, cascaded up to the enterprise level.

$\mathbf{R}^{Eng}$: At iteration 1 of the ATC formulation [72, 77] $\mathbf{R}^{Eng}$ represents the enterprise estimation of engineering design capabilities. This will be updated with each iteration to reflect the true design values achievable by the engineering level, i.e. $\mathbf{R}^{Eng^L}$.

$Cost^L_{Architecture\{j\}}$: This represents the product cost based on the engineering capabilities of meeting predicted customer attributes. At iteration 1, this is estimated by enterprise decision makers and updated to reflect the true cost based on engineering response thereafter.

$\pi_{Architecture_j}$: Profit of architecture j, which is a function of price and cost of the product variant j.

$\varepsilon_{\mathbf{R}}$: Deviation tolerance between customer performance & targets and engineering response.

$\varepsilon_{\mathbf{C}}$: Deviation tolerance between enterprise product cost estimation & targets and engineering response.

Engineering Level

This is where the individual architecture costs are calculated, along with the physical product architecture design. The engineering design level is modeled as a mixed integer nonlinear programming problem [120], with discrete

selection variables that govern component choice selections (manufacturer specifications, component design, etc) and continuous variables that regulate the product dimensions and aesthetic design. The iteration between the *enterprise level* and the *engineering level* determines the feasibility criteria for each product as customer targets are set at the enterprise level and subsequently validated with an engineering subsystem response within a specified tolerance of ε. Mathematically, this is represented as:

Given

$\mathbf{R}^{Eng^U}$

$$\min \; Cost_{Architecture_j} + \|\mathbf{R}^{Eng^U} - \mathbf{R}^{Eng}\|_2^2 \tag{4.13}$$

with respect to

$\mathbf{x}_{Eng}$

subject to:

Engineering product design equality constraints,

production capacity, materials, supplier constraints

$$\mathbf{h}_{Eng}(\mathbf{x}_{Eng}) = \mathbf{0} \tag{4.14}$$

$$\mathbf{g}_{Eng}(\mathbf{x}_{Eng}) \leq \mathbf{0} \tag{4.15}$$

Engineering Level: Variable Notation Definitions

$Cost_{Architecture_j}$: The engineering design objective, *Cost*, is the primary performance criterion influencing the product design, while the objective is not limited to the cost. The objective can be any individual product performance objective, such as cost, weight, etc. In the cell phone example problem, the engineering objective is to minimize the cost as well as matching the attributes targets $\mathbf{R}^{Eng^U}$.

$\mathbf{R}^{Eng^U}$: Engineering performance response target from the enterprise system level, cascaded down to the engineering level.

$\mathbf{R}^{Eng}$: Performance response from the engineering design, i.e., $\mathbf{R}^{Eng}$=$\mathbf{R}^{Eng}(\mathbf{x}_{Eng})$,

(The engineering response $\mathbf{R}^{Eng}$ will become $\mathbf{R}^{Eng^L}$ at the enterprise system level.)

The *product architecture* is defined in this work as the engineering design foundation from which product variants can evolve. The functionality of each product architecture is unique and addresses the fundamental requirements of

the product. For example, an MP3 product architecture would be designed such that the MP3 functionality can be easily accessed and controlled by the user. An interactive Game capable (Noted as *Games* in the cell phone example) cell phone would have a product architecture that allows the user to seamlessly switch from Game playing mode to phone operation mode. These differences are addressed in the engineering design level, where customer attributes are translated into engineering design functionality through a set of linear constraints (See Figure 4.3 and Figure 4.2).

Enterprise Portfolio Selection

This is where the overall enterprise portfolio profit is determined by searching through the feasible product space and selecting/deselecting architectures in an attempt to maximize profit by generating an optimal product portfolio. Here, the optimal portfolio is defined as the selected products that maximize the enterprise profit within the product portfolio limit K. The termination of this selection process is determined when either 1) the product portfolio limit is reached in case there exist more profitable product concepts than the limit, or 2) all the profitable product concepts are identified in case the number of profitable product concepts is less than the limit. Mathematically, this is represented as:

$$\min \; -\sum_{j=1}^{k} x_j \cdot \pi_{Architecture(j)} \tag{4.16}$$

subject to

$$h1 : x_j = \{0,1\} \quad j \in \{1,\ldots,k\} \tag{4.17}$$

$$g1 : \sum_{j=0}^{k} x_j - K \quad \leq 0 \tag{4.18}$$

Enterprise Portfolio Selection: Variable Notation Definitions

$\pi_{Architecture(j)}$: Profit of architecture j

x_j : Binary variable selecting or deselecting particular architecture ($\pi_{Architecture}$) where $\sum_{j=1}^{k} x_j \leq K$

k**:** *Total feasible product/variants* that can be designed. This numeric value is attained through the engineering design validation process. The value k therefore represents the total number of product/variants that satisfy customer performance and price expectations

K**:** *Product portfolio limit*. To avoid impractical manufacturing expectations and an over-saturation of products in the market space, the number of products existing in the product portfolio must be constrained. The value set as the maximum portfolio limit may be a function of many externalities including competition, distribution and marketing constraints, etc. In the proposed approach, the product portfolio limit is assumed to be set by the enterprise decision maker. (**Note:** Depending on the number of existing feasible products, this limit may/may not be reached.)

The flow diagram in Figure 4.1 represents the overall process from customer preference acquisition via database extraction to the generation of product concepts. The validated product concepts with the highest profit margins will form the product portfolio (Subject to the product portfolio limit as determined by enterprise decision makers).

4.2 Application: Cell Phone Design

4.2.1 PHASE 1: Cell Phone Customer Knowledge Discovery

To validate the proposed decision tree approach in generating a product portfolio, a cell phone product portfolio case study is presented. A cell phone survey was designed using the UIUC webtools platform where respondents had the option of selecting a combination of attribute values and the price category that most closely represented their selection [121] . To emphasize the strength of data mining in handling large datasets, additional data was simulated (based on the generated survey questionnaire) using Excel Visual Basic to achieve a total of 40,000 customer responses. The data preprocessing steps explained in section 1.4.1 are handled by the data mining analysis tool [1]. In machine learning techniques, the raw data is partitioned: typically 2/3 is used to train the algorithm and the remaining 1/3 to test the model for predictive accuracy [1]. For demonstration purposes, a small fraction of the train data T is used to illustrate the decision tree generation algorithm discussed earlier. A set of 10 cases will demonstrate the gain ratio criteria in decision tree decomposition (see Table 4.2).

The class variable in Table 4.2 is *MaxPrice* and is defined as the maximum price a customer is willing to pay

for a particular product. The class variable can be altered, depending on the focus of the enterprise decision makers to reflect the strategic objectives of the company. The proposed methodology aims to quantify the price sensitivity information predicted by the decision tree model.

Each row in Table 4.2 will be defined as an independent *Case*. (The term *Case* refers to a unique customer response containing certain attribute values along with the associated class value). There are 6 attributes in the example Table represented as {Feature, Priority, Type, Connectivity, Battery Life, and Display}. The class variable MaxPrice is partitioned into 5 separate, mutually exclusive classes [$40, $80, $120, $160, $200].

Since the 10 cases in the example do not all belong to the same class, the *C4.5 divide and conquer* algorithm can be implemented in an attempt to split the cases into subsets. There are 4 classes in the cell phone sample train ***T*** file (The $40 class of MaxPrice did not occur in this illustration but does in larger training sets). ***T*** contains three cases belonging to $200 class, four cases belonging to $160 class, two cases belonging to $120 class and one case belonging to $80 class for a total of 10 cases for the training data in Table 4.2.

Product Concept Generation Through C4.5 Decision Tree Classification

Step 1 *Class Identification*

Following the C4.5 algorithm, the first step is to determine the average information needed to identify a value of MaxPrice in the training data. *info*(*T*)[bit] will be defined as:

$$info(T) = -\frac{3}{10}\cdot\log_2(\frac{3}{10}) - \frac{4}{10}\cdot\log_2(\frac{4}{10}) - \frac{2}{10}\cdot\log_2(\frac{2}{10}) - \frac{1}{10}\cdot\log_2(\frac{1}{10}) = 1.846[bits] \quad (4.19)$$

The above *info(T)* calculation is determined directly from Table 4.2 where the information needed to identify the three cases of the $200 class out of the total 10 cases is represented in equation (4.19) as $-\frac{3}{10}\cdot\log_2(\frac{3}{10})$, and similarly for each subsequent class identification.

Step 2 *Attribute Selection*

The information gained by selecting a particular attribute will determine the sequence of attribute selection and consequently the structure and length of the decision tree or in product development terms, the number of candidate product concepts that are generated and deemed to be the best predictors of each class of MaxPrice. The tree decomposition process is an iterative approach, substituting one attribute over another if a higher information gain can be realized by selecting this attribute as a node in the tree. To demonstrate this process, an attribute is selected as the initial node (root) with the information gained calculated for the given selection.

(Attribute Test=Feature) The attribute selected is then partitioned into its individual mutually exclusive outcomes

Table 4.2: Test Data for Decision Tree Generation

Feature	Objective	Design	Connectivity	Battery Life	Display	MaxPrice	
MP3	Cost	Flip	Bluetooth	5	ScreenSize	200	Branch 1
MP3	Weight	Flip	Wifi	3	ScreenSize	160	
MP3	Weight	Flip	Bluetooth	3	Resolution	160	
MP3	Cost	Shell	Infrared	5	ScreenSize	80	
Camera	Weight	Shell	Wifi	3	ScreenSize	120	Branch 2
Camera	Weight	Flip	Wifi	3	Resolution	120	
Games	Cost	Flip	Bluetooth	5	Resolution	200	Branch 3
Games	Cost	Shell	Wifi	7	Resolution	200	
Games	Weight	Flip	Infrared	5	ScreenSize	160	
Games	Cost	Flip	Bluetooth	3	ScreenSize	160	

(represented by branches in the actual decision tree). There are 4 cases that are MP3, 2 cases that are Camera, and 4 cases that are Games to comprise of the 10 Feature cases as illustrated in Figure 4.2. The expected information requirement of the *Feature* attribute is the weighted sum of the three subsets {MP3, Camera, Games}.

$$\begin{aligned} info_{\{X=Feature\}}(\tau) = \frac{4}{10}\cdot\{-\frac{1}{4}\cdot\log_2(\frac{1}{4})-\frac{2}{4}\cdot\log_2(\frac{2}{4}) \\ -\frac{0}{4}\cdot\log_2(\frac{0}{4})-\frac{1}{4}\cdot\log_2(\frac{1}{4})-\frac{0}{4}\cdot\log_2(\frac{0}{4})\} \\ +\frac{2}{10}\cdot\{-\frac{0}{2}\cdot\log_2(\frac{0}{2})-\frac{0}{2}\cdot\log_2(\frac{0}{2}) \\ -\frac{2}{2}\cdot\log_2(\frac{2}{2})-\frac{0}{2}\cdot\log_2(\frac{0}{2})-\frac{0}{2}\cdot\log_2(\frac{0}{2})\} \\ +\frac{4}{10}\cdot\{-\frac{2}{4}\cdot\log_2(\frac{2}{4})-\frac{2}{4}\cdot\log_2(\frac{2}{4}) \\ -\frac{0}{4}\cdot\log_2(\frac{0}{4})-\frac{0}{4}\cdot\log_2(\frac{0}{4})\}-\frac{0}{4}\cdot\log_2(\frac{0}{4}) \\ = 1.00 \ \ bits \end{aligned} \tag{4.20}$$

Therefore, the information gained by testing $Attribute = Feature$ is simply:

$$\begin{aligned} gain(X) &= info(T) - info_{\{X=Feature\}} \\ &= 1.864 - 1.00 = 0.864 \ [bits] \end{aligned} \tag{4.21}$$

In the event that the data set contains one or several attributes with a significantly greater range of outcomes, the Split Info(X) function can attempt to normalize the attributes.

$$GainRatio(X) = \frac{gain(X)}{SplitInfo(X)} = 0.57 \tag{4.22}$$

Each subsequent attribute that is tested on the basis of *gain ratio* criteria is compared with the previous attribute, and substituted if a higher gain ratio is achieved. This iterative process is continued until a single class is identified for a given attribute split.

Translation of customer attributes to engineering design functionality Customer predicted attribute information must be translated into meaningful engineering functionality criterion for the product design process. A set of linear equations represented by Figure 4.2 indicate which of the product functionality components are included in a particular product architecture. Figure 4.3 is simply a textual explanation of the A-matrix and indicates which of the engineering components are active.

Depending on the cell phone architecture type and the engineer design objective function, one or several of the elements in each row of the A-matrix will be active (1) or inactive (0). The upper and lower bounds for the linear equations (b-matrix) therefore fluctuate based on the product concept requirements currently being tested. For example, if an MP3 product concept requires a *bluetooth* connectivity feature, the element representing bluetooth connectivity in row 7 of the A-matrix will automatically be active (1) and the lower bound for the connectivity linear constraint (which comprises of 3 possible connectivity options: Bluetooth, Infrared or Wifi (See row 7 of Figure 4.3)) will immediately be set to 1. That is, b_7 of Figure 4.2 will be ≥ 1. Furthermore, the lower bound for the external speaker (Row 11 of the A-matrix in Figure 4.3) will be set to 1, indicating that the MP3 cell phone, will come equipped with external audio capability (A functionality translation based on the customer attribute requirement of MP3 music playback). The red highlighted numbers in each row of the A-matrix in Figure 4.3 (i.e., column indices) indicate the number of possible choices for that particular component group.

4.2.2 PHASE 2: Product Concept Validation

Enterprise Level: Cell Phone Design Validation and Profit Calculation Once the customer data set of 40,000 cases (with 576 unique attribute combinations) has been narrowed down to 46 generated product concepts (vector of

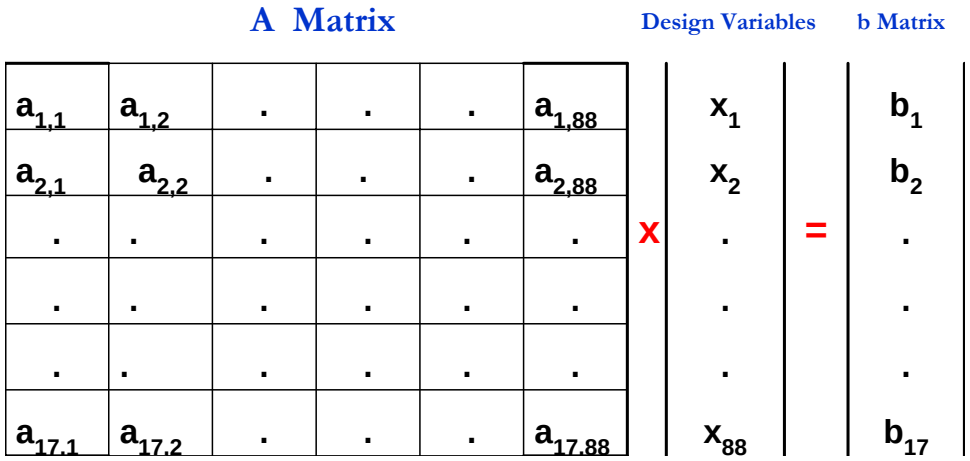


Figure 4.2: Set of linear design equations (in matrix form) guiding the product architecture formulation

predicted product attribute combinations) via the C4.5 data mining tree generation technique, the engineering design feasibility and potential profit margin for each product can now be determined: mathematically represented as follows: Given

$$T^{BatteryLife}, T^{Connection}, T^{Priority}, T^{Display}, T^{Type}$$

$$MaxPrice_j, d_j, R^{BatteryLife^{Eng^L}}, R^{Connection^{Eng^L}}$$

$$R^{Priority^{Eng^L}}, R^{Display^{Eng^L}}, R^{Type^{Eng^L}}, Cost_j^L$$

$$\begin{aligned} \min \; & -\pi_{Architecture_j} + \|T^{BatteryLife} - R^{BatteryLife^{Eng}}\|_2^2 \\ & + \|T^{Connection} - R^{Connection^{Eng}}\|_2^2 + \|T^{Priority} - R^{Priority^{Eng}}\|_2^2 \\ & + \|T^{Display} - R^{Display^{Eng}}\|_2^2 + \|T^{Type} - R^{Type^{Eng}}\|_2^2 \\ & + \varepsilon_{BatteryLife} + \varepsilon_{Connection} + \varepsilon_{Priority} \\ & + \varepsilon_{Display} + \varepsilon_{Type} + \varepsilon_C \end{aligned} \tag{4.23}$$

with respect to

$$R^{BatteryLife^{Eng}}, R^{Connection^{Eng}}, R^{Priority^{Eng}},$$

$$R^{Display^{Eng}}, R^{Type^{Eng}}, Cost_j$$

Linear Constraints: Cell Phone Engineering Design	
A Matrix (Functional Component Selection Process)	**Matrix Element Value**
A(1,1:3) Cell Phone 32MB RAM: 3 Manufacturers to choose from	1
A(2,4:6) Cell Phone 64MB RAM: 3 Manufacturers to choose from	1
A(3,7:12) Cell Phone External Memory Storage: 6 Manufacturers to choose from	1
A(4,13:18) Cell Phone Hard Drive Storage: 3 manufacturers of 1 Gig and 3 manufactures of 2 Gig	1
A(5,[19:25]) Cell Phone Design: 2 Designs (Flip phone or Shell phone) manufactured in-house	1
A(6,[31:38]) Cell Phone Battery Design: 2 Types (NIMH or LION) manufactured in-house	1
A(7,[46,47,48]) Cell Phone Connectivity: Bluetooth, Wifi, Infrared from manufacturer	1
A(8,[49,50]) Cell Phone Microphone selection: 2 Manufacturers to choose from	1
A(9,[51,52,53]) Cell Phone Earpiece: 3 Manufacturers to choose from	1
A(10,[54,55]) Cell Phone Audio: Audio Jack component- 2 manufacturers to choose from	1
A(11,[56,57,58]) Cell Phone External Audio: External Speaker- 3 manufacturers to choose from	1
A(12,[59:66]]) Cell Phone Display Type: 2 Designs (TFT or OLED display) manufactured in-house	1
A(13,[73,74,75,76]]) Cell Phone Camera module: 4 manufacturers to choose from (1Mpixel VS 2 Mpixel)	1
A(14,[77,78,79,80]]) Cell Phone MP3 module: 4 manufacturers to choose from	1
A(15,[81,82]]) Cell Phone Internet module: 2 manufacturers to choose from	1
A(16,[83,84,85]]) Cell Phone processor for Games capability: 3 manufacturers to choose from	1
A(17,86) Cell Phone module for SMSText capability:1 manufacturer to choose from	1

Figure 4.3: A matrix forming the linear equation set. The matrix is sparse, with active elements signified by a value of 1

subject to

$$h1 : \pi_{Architecture_j} - d_j \cdot (MaxPrice_j - Cost_j) = 0 \tag{4.24}$$

$$h2 : MaxPrice_j = \{\$40, \$80, \$120, \$160, \$200\} \tag{4.25}$$

$$g1 : \|R^{BatteryLife^{Eng}} - R^{BatteryLife^{Eng^L}}\|_2^2 \leq \varepsilon_{BatteryLife} \tag{4.26}$$

$$g2 : \|R^{Connection^{Eng}} - R^{Connection^{Eng^L}}\|_2^2 \leq \varepsilon_{Connection} \tag{4.27}$$

$$g3 : \|R^{Priority^{Eng}} - R^{Priority^{Eng^L}}\|_2^2 \leq \varepsilon_{Priority} \tag{4.28}$$

$$g4 : \|R^{Display^{Eng}} - R^{Display^{Eng^L}}\|_2^2 \leq \varepsilon_{Display} \tag{4.29}$$

$$g5 : \|R^{Type^{Eng}} - R^{Type^{Eng^L}}\|_2^2 \leq \varepsilon_{Type} \tag{4.30}$$

$$g6 : \|Cost_j - Cost_j^L\|_2^2 \leq \varepsilon_C \tag{4.31}$$

Here, the attributes are given as product design targets **T** and the engineering design responses are **R**, for which deviations are defined as ε. Individual product demand is noted d_j with corresponding price $MaxPrice_j$ and cost $Cost_j$. The initial evaluation of the engineering design response is estimated and then subsequently updated with each engineering design response thereafter.

Engineering Level: Product Design Validation After the enterprise profit is calculated for each of the 46 product variant concepts, individual product variants are checked for their feasibility in the engineering design space. Based on the attribute targets, the engineering design team attempts to minimize the cost while meeting the product attribute requirements.

Given

$$R^{BatteryLife^U}, R^{Connection^U}, R^{Priority^U}, R^{Display^U}, R^{Type^U}$$

$$\begin{aligned} \min\ & Cost_{Architecture_j} + \|R^{BatteryLife^U} - R^{BatteryLife}\|_2^2 \\ & + \|R^{Connection^U} - R^{Connection}\|_2^2 + \|R^{Priority^U} - R^{Priority}\|_2^2 \\ & + \|R^{Display^U} - R^{Display}\|_2^2 + \|R^{Type^U} - R^{Type}\|_2^2 \end{aligned} \tag{4.32}$$

with respect to

$$\mathbf{x}_{Eng}$$

subject to[1]

$$\begin{aligned} &\text{Screen Resolution Constraints,} \\ &\text{Battery Design Constraints,} \\ &\text{Outer Casing Design (Phone Type) Constraints,} \\ &\text{Design Priority Constraints,} \\ &\mathbf{g}_{Eng}(\mathbf{x}_{Eng}) \leq \mathbf{0}, \mathbf{h}_{Eng}(\mathbf{x}_{Eng}) = \mathbf{0} \end{aligned} \tag{4.33}$$

Product Portfolio Selection Among the feasible product variants (35 out of 46), the final step is to generate product portfolio under the specified limit of 7 total products. For each product variant, the selection variable $\mathbf{x}$ is defined to achieve the final most profitable product portfolio.

$$\min\ -\sum_{j=0}^{k} x_j \cdot \pi_{Architecture(j)} \tag{4.34}$$

subject to

$$h1 : x_j = \{0,1\} \quad j \in \{1,\ldots,35\} \tag{4.35}$$

$$g1 : \sum_{j=0}^{35} x_j - 7 \quad \leq 0 \tag{4.36}$$

[1]To enhance the overall flow, the elaborate constraints governing the engineering design of cell product variants are condensed and represented by only $\mathbf{g}_{Eng}(\mathbf{x}_{Eng})$ and $\mathbf{h}_{Eng}(\mathbf{x}_{Eng})$ above. Refer to the Appendix for detailed cell phone design model.

4.3 Results and Discussion

Table 4.3: Results of C4.5 data mining product concept generation. The yellow highlighted rows indicate members of the optimal product portfolio.

Product Platform	Product Variants	Objective	Design	Connectivity	Battery_Life	Display	Demand	MaxPrice	Engineering Design Validation	Product Unit Cost	Generated Profit	Product Portfolio Member
Generic	1	Weight		Bluetooth	3		293	$80	Feasible	$46.04	$9,951	Yes
	2	Cost		Bluetooth	3		297	$40	Feasible	$43.66	-$1,087	No
	3			Infrared	3		591	$80	Feasible	$41.59	$22,701	Yes
	4	Weight		Wifi	3		284	$40	Feasible	$47.77	-$2,206	No
	5	Cost		Wifi	3		318	$80	Feasible	$45.42	$10,997	Yes
	6	Weight		Bluetooth	5		290	$40	Feasible	$53.69	-$3,970	No
	7	Weight		Infrared	5		283	$40	Feasible	$57.08	-$4,833	No
	8	Weight		Wifi	5		302	$80	Feasible	$60.65	$5,845	Yes
	9	Cost	Flip		5		468	$80	Feasible	$45.04	$16,363	Yes
	10	Cost	Shell		5		413	$40	Feasible	$51.28	-$4,659	No
	11				7		1123	$40	Infeasible	$53.73	-$15,422	No
SMS Text	1				3	ScreenSize	907	$160	Feasible	$45.61	$103,752	Yes
	2		Flip		3	Resolution	441	$160	Feasible	$48.44	$49,196	Yes
	3		Shell		3	Resolution	423	$80	Feasible	$47.91	$13,575	Yes
	4	Weight		Bluetooth	5		314	$160	Feasible	$66.64	$29,316	Yes
	5	Cost		Bluetooth	5		331	$80	Feasible	$58.33	$7,174	Yes
	6			Infrared	5		578	$120	Feasible	$56.26	$36,844	Yes
	7			Wifi	5		600	$80	Feasible	$59.83	$12,104	Yes
	8				7	ScreenSize	579	$80	Infeasible	$61.22	$10,872	No
	9	Weight			7	Resolution	296	$80	Infeasible	$64.09	$4,710	No
	10	Cost			7	Resolution	294	$40	Infeasible	$64.06	-$7,073	No
Games	1			Bluetooth	3		581	$160	Feasible	$55.33	$60,812	Yes
	2			Bluetooth	5		563	$120	Feasible	$68.21	$29,158	Yes
	3			Infrared			1185	$160	Feasible	$49.42	$131,031	Yes
	4			None			1104	$120	Feasible	$45.69	$82,033	Yes
	5	Weight		Wifi			598	$160	Feasible	$56.30	$62,011	Yes
	6	Cost		Wifi	3		304	$120	Feasible	$56.83	$19,203	Yes
	7	Cost		Wifi	5		321	$160	Feasible	$69.71	$28,983	Yes
Camera	1			Bluetooth			1166	$200	Feasible	$87.59	$131,075	Yes
	2			Infrared			1222	$200	Feasible	$88.58	$136,150	Yes
	3			None		ScreenSize	602	$120	Feasible	$88.59	$18,909	Yes
	4			None		Resolution	580	$80	Feasible	$79.99	$6	No
	5			Wifi			1184	$200	Feasible	$79.98	$142,103	Yes
Internet	1			Bluetooth		ScreenSize	583	$120	Feasible	$54.67	$38,085	Yes
	2			Bluetooth		Resolution	546	$160	Feasible	$57.51	$55,960	Yes
	3	Weight		Infrared			559	$160	Feasible	$56.59	$57,807	Yes
	4	Cost		Infrared			543	$120	Feasible	$52.60	$36,596	Yes
	5	Weight	Flip	None			295	$160	Feasible	$52.86	$31,607	Yes
	6	Cost	Flip	None			294	$80	Feasible	$48.87	$9,151	Yes
	7	Weight	Shell	None			297	$80	Feasible	$51.53	$8,455	Yes
	8	Cost	Shell	None			295	$160	Feasible	$48.36	$32,932	Yes
	9			Wifi			1120	$120	Feasible	$56.17	$71,485	Yes
MP3	1	-	-	Bluetooth	-	-	1239	$200	Feasible	$98.48	$125,778	Yes
	2	-	-	Infrared	-	-	1108	$200	Feasible	$95.90	$115,337	Yes
	3	-	-	None	-	-	1124	$80	Feasible	$92.17	-$13,685	No
	4	-	-	Wifi	-	-	1161	$200	Feasible	$99.47	$116,710	Yes

The proposed product portfolio methodology presents more than just a set of feasible product concepts, but rather a validated portfolio of product designs that are the best indicators of market success which ultimately maximize overall enterprise profit. Table 4.3 presents the final solution achieved in the cell phone case study of 40,000 customer responses that are subsequently narrowed down to 46 predictive product concepts. As can be seen in Table 4.3 , column 10, the multilevel optimization formulation returns a vector of feasible/infeasible product designs based on customer predictive preference targets cascaded down to the engineering level. In the proposed methodology, the term *feasibility* is defined as customer preference targets attained through data mining predictive techniques that are matched within the engineering design response tolerance of (ε=0.01). A product design that fails to satisfy this tolerance is considered to be a suboptimal product variant and is excluded in the optimal product portfolio.

Table 4.4: Detailed Engineering Design Product Variant Solutions

Variable Description	Component Source	SMS Phone1 Solution	SMS Phone2 Solution	Phone3 Solution	SMS Phone4 Solution	Phone5 Solution	SMS Phone6 Solution	Phone7 Solution	SMS Phone8 Solution	Phone9 Solution	SMS Phone10 Solution	Units
Objective		Cost	Cost	Cost	Weight	Cost	Cost	Cost	Cost	Weight	Cost	
64 Megabyte Discrete	Manufacturer 1	0	0	0	1	0	0	0	0	1	0	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	0	-
	Manufacturer 3	1	1	1	0	1	1	1	1	0	1	-
Shell Phone Discrete	Engineering Design	0	0	1	1	0	0	0	0	0	0	-
Phone Length	Engineering Design	80.5	120.0	80.0	80.0	120.0	120.0	120.0	120.0	120.0	120.0	mm
Phone Width	Engineering Design	40.0	40.0	40.0	40.0	40.0	40.0	40.0	46.5	40.0	40.0	mm
Phone Thickness	Engineering Design	12.0	16.3	13.1	22.4	16.3	12.0	12.0	12.0	16.3	16.3	mm
Phone Weight	Engineering Design	19.7	40.0	21.4	36.5	40.0	29.4	29.4	34.1	40.0	40.0	g
Phone Cost	Engineering Design	8.8	18.0	9.6	16.4	18.0	13.2	13.2	15.3	18.0	18.0	$
Flip Phone Discrete	Engineering Design	1	1	0	0	1	1	1	1	1	1	-
Phone Length	Engineering Design	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	mm
Phone Width	Engineering Design	45.0	45.0	45.0	45.0	45.0	45.0	45.0	45.0	45.0	45.0	mm
Phone Thickness	Engineering Design	12.0	12.0	18.1	18.1	13.2	13.2	13.2	18.1	18.1	18.1	mm
Phone Weight	Engineering Design	26.5	26.5	40.0	40.0	29.2	29.2	29.2	40.0	40.0	40.0	g
Phone Cost	Engineering Design	7.9	7.9	12.0	12.0	8.8	8.8	8.8	12.0	12.0	12.0	$
Nickel Metal Hydride	Engineering Design	0	0	0	0	0	0	0	0	0	0	-
Battery Weight	Engineering Design	50.0	50.0	50.0	49.6	50.0	50.0	50.0	50.0	50.0	50.0	g
Battery Length	Engineering Design	80.0	60.6	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	mm
Battery Width	Engineering Design	22.2	60.0	48.8	26.4	22.2	22.2	22.2	22.2	48.8	48.8	mm
Battery Thickness	Engineering Design	30.0	14.7	13.7	25.0	30.0	30.0	30.0	30.0	13.7	13.7	mm
Battery Cost	Engineering Design	20.2	20.2	20.2	20.0	20.2	20.2	20.2	20.2	20.2	20.2	$
Battery Capacity	Engineering Design	1139.4	1139.4	1139.4	1129.4	1139.4	1139.4	1139.4	1139.4	1139.4	1139.4	mAh
Lithium Ion Battery	Engineering Design	1	1	1	1	1	1	1	1	1	1	-
Battery Weight	Engineering Design	9.49	9.49	9.49	16.20	16.19	16.19	16.19	22.19	22.19	22.19	g
Battery Length	Engineering Design	48.30	46.56	48.00	48.00	72.00	72.00	72.00	72.00	72.00	72.00	mm
Battery Width	Engineering Design	41.21	42.75	38.00	38.00	42.75	42.75	42.75	42.75	42.75	42.75	mm
Battery Thickness	Engineering Design	5.40	5.40	5.89	10.06	5.96	5.96	5.96	8.16	8.16	8.16	mm
Battery Cost	Engineering Design	8.64	8.64	8.63	14.74	14.73	14.73	14.73	20.19	20.19	20.19	$
Battery Capacity	Engineering Design	464.23	464.23	463.94	792.34	791.95	791.95	791.94	1085.13	1085.13	1085.13	mAh
Cell Phone Talk Time	Engineering Design	3.00	3.00	3.00	5.00	5.00	5.00	5.00	6.79	6.79	6.79	h
Bluetooth Discrete	Manufacturer 1	0	0	0	1	1	0	0	0	0	0	-
WiFi Discrete Variable	Manufacturer 1	0	0	0	0	0	0	1	0	0	0	-
Infra-Red Discrete	Manufacturer 1	0	0	0	0	0	1	0	0	0	0	-
Microphone Discrete	Manufacturer 1	1	1	1	1	1	1	1	1	1	1	-
	Manufacturer 2	0	0	0	0	0	0	0	0	0	0	-
EarPiece Discrete	Manufacturer 1	1	1	1	0	1	1	1	1	0	1	-
	Manufacturer 2	0	0	0	1	0	0	0	0	1	0	-
	Manufacturer 3	0	0	0	0	0	0	0	0	0	0	-
LCD Discrete Variable	Engineering Design	1	0	0	0	1	1	1	1	0	0	-
Length of LCD	Engineering Design	30.00	30.00	24.00	35.71	30.00	30.00	30.00	30.00	30.00	30.00	mm
Width of LCD	Engineering Design	31.50	31.50	28.00	28.00	31.50	31.50	31.50	31.50	33.33	31.50	mm
Display Resolution	Engineering Design	13929.30	13929.30	9906.16	14740.00	13929.30	13929.30	13929.30	13929.30	14740.00	13929.30	pixels
LCD Manufacturing	Engineering Design	4.73	4.73	3.36	5.00	4.73	4.73	4.73	4.73	5.00	4.73	$
LCD Unit Weight	Engineering Design	37.80	37.80	26.88	40.00	37.80	37.80	37.80	37.80	40.00	37.80	g
LCD Power	Engineering Design	9.45	9.45	6.72	10.00	9.45	9.45	9.45	9.45	10.00	9.45	mhA
OLED Discrete	Engineering Design	0	1	1	1	0	0	0	0	1	1	-
Length of OLED	Engineering Design	30.00	30.00	24.00	24.00	30.00	30.00	30.00	30.00	30.00	30.00	mm
Width of OLED	Engineering Design	31.50	31.50	28.00	28.00	31.50	31.50	31.50	31.50	31.50	31.50	mm
Display Resolution	Engineering Design	18540.90	18540.90	13184.64	13184.64	18540.90	18540.90	18540.90	18540.90	18540.90	18540.90	pixels
OLED Manufacturing	Engineering Design	7.56	7.56	5.38	5.38	7.56	7.56	7.56	7.56	7.56	7.56	$
OLED Unit Weight	Engineering Design	28.35	28.35	20.16	20.16	28.35	28.35	28.35	28.35	28.35	28.35	g
Consumption	Engineering Design	28.35	28.35	20.16	20.16	28.35	28.35	28.35	28.35	28.35	28.35	MhA
SMS Module	Manufacturer 1	1	1	1	1	1	1	1	1	1	1	-
Product Unit Cost	**Solution**	**45.61**	**48.44**	**47.91**	**66.64**	**58.33**	**56.26**	**59.83**	**61.22**	**64.09**	**64.06**	**$/unit**
Product Unit Weight	**Solution**	**93.35**	**83.90**	**70.61**	**99.94**	**111.69**	**111.29**	**115.58**	**119.59**	**108.74**	**110.14**	**g**

Product feasibility is however not the only measure of product design success. With the incorporation of demand information directly acquired through the C4.5 data mining process, each product variant profit can be calculated based on the unit product cost, the MaxPrice class prediction and the demand for a particular product concept *i*. Referring to the results for the **Generic Phone** architecture in Table 4.3, there are 11 product concepts generated through the data mining technique. As the results indicate, generic product variant **11** with a predicted battery life expectation of **7 hours** and a MaxPrice prediction of **$40**, was found to be infeasible in the engineering design formulation. The violated target in this scenario was that of the battery life with a maximum attainable engineering design response of **6.79 hours** (Table 4.4). Using the proposed metric for evaluating feasible designs, this product concept clearly violates the tolerance limit, hence is excluded as a candidate for the optimal product portfolio. In addition to the feasibility check, the proposed approach also generates the unit cost for product design with its corresponding profit. For this specific variant, Generic variant 11 in the Table 4.3, the unit cost is $53.73, therefore its corresponding loss is $15,422. Customers shows their preference for this specific variant. However, this concept should not be pursued due to the projected loss.

Referring to the Generic architecture results in Table 4.3, there are several product concepts that have a feasible engineering design but are omitted in the optimal product portfolio set. As discussed earlier, this is due to the fact that *overall enterprise profit* is the second criteria for evaluating product variants to be included in the optimal product portfolio. There are total of 11 infeasible and/or negative profit generating product variants out of the 46 product concepts predicted by the data mining process, resulting in 35 candidate products to introduce to the cell phone market. Depending on the enterprise product portfolio limit, and the number of product families that can be managed, all or a few of these products could be considered for market launch.

If a maximum portfolio limit size of 7 is assumed, the enterprise optimal product portfolio as described in section 4.1.3, would simply be a selection of the most profitable product variants, subject to the portfolio limit constraint. This is modeled by the selection problem in (4.34) – (4.36). Exploring the entire product concept space generates a solution of 7 product variants spanning multiple product families. The final solution yields 1 product variant from the Games product family, 3 product variants from the Camera product family (Camera product variants $\{1,2,5\}$) and 3 product variants from the MP3 product family (MP3 product variants $\{1,2,4\}$) yielding a total product portfolio sales volume (based on demand information) of **8265 units** and an overall enterprise profit of **$898,185**.

Such powerful insights enable enterprise decision makers to evaluate products/variants based on several dimensions of performance. In the example of 40,000 customers, each customer does not have to be provided with his/her own unique customizable product, but rather purchasing behaviors can be addressed with the 46 product concepts generated in the data mining predictions. Furthermore, enterprise decision makers can determine which out of these product concepts would be the most successful in an attempt to maximize profit.

4.4 Conclusion

The volatility of highly competitive consumer markets is the major driving force shaping company strategies in product development. The power to accurately predict and design products before they launch is a fundamental tool in ensuring a competitive advantage amongst fierce competition. The major focus of this research is to predict customer wants and subsequently design the most profitable products or product variants. The predictive aspects of product development are addressed through data mining techniques that generate candidate product concepts along with individual predicted demand information. The validation of these product concepts at the engineering design level increases the likelihood of the products being market successes if launched. As a result, enterprise decision makers will have several options in formulating an optimal product portfolio. Other metrics such as level of commonality among product variants can be used as an additional evaluation metric in deciding product launches. Additional cost savings benefits may be realized through post optimality analysis of shared components.

4.5 Product Concept Generation with Multi-Response Preference Data

Engineering design efforts often involve many disciplines trying to coordinate between system performance expectations and design capabilities. Product performance expectations can include many competing objectives such as *high reliability*, *low energy consumption*, *minimal environmental impact*, to name but a few. In today's modern era of mass customization, design engineers are faced with the challenge of delivering product diversity at a relatively low cost. There have been several proposed techniques for acquiring and incorporating performance feedback into the systems design process, ranging from traditional statistical driven methods such as conjoint analysis and discrete choice analysis to data mining methods such as decision tree classification and clustering approaches [21, 22, 55, 122]. A major drawback of existing performance preference acquisition techniques is their limitation to single output response variable predictions. Single output response variable predictions may be insufficient for design teams in realistic product development settings when attempting to accurately model next generation designs.

To overcome some of the limitations of single response modeling techniques, this section presents a novel approach to predict multiple output responses. The proposed Multi-Response Data Mining methodology is a tree induction algorithm that attempts to simultaneously predict multiple variable responses (performance criterion) based on an output response filtering technique and a generalized association measure for efficient model generation. The results of the Multi-Response Data Mining model will serve as system performance targets in a multi-objective design space and aid in assessing the feasibility of product design solutions. Instead of exhaustively investigating the Pareto space in a multi-objective design model, design engineers will be guided by the performance target solutions which will serve as product design targets in the multi-objective design model.

4.5.1 Existing Single Response Predictive Data Mining

There are many well established predictive data mining algorithms capable of handling single response predictions with proven results. Some of which include Naïve Bayes classification, Support Vector Machines, Decision Tree Classification, to name but a few [95].

The Naïve Bayesian data mining algorithm assigns output response predictions by employing Bayes' Theorem in calculating *posterior probabilities*. The fundamental assumption of the Naïve Bayes algorithm is the assumption of attribute independence which significantly reduces the complexity of the formulation. Under the Maximum Likelihood function, the correct output response prediction is achieved so long as it is more probable than any other output response [80].

Support Vector Machines (SVMs) is a relatively new classification algorithm that has been successfully applied to both linear and non-linear data sets. SVMs attempt to identify the unique response values by creating the maximum-separating hyper-plane that partitions the instances within the data set that belong to a particular response. By transforming the original data into a higher order dimensionality space, SVMs can identify the optimal boundary that maximizes the distance between the output response labels and the hyper-plane [123].

The Decision Tree Classification algorithm predicts the response variable based on a data-partitioning strategy that incorporates the notion of *node impurity* in determining the structure of the tree. Attributes are independently tested to determine their ability to reduce the impurity of the response variable at each node of the tree. The iterative data partitioning sequence continues until a single response is identified, accompanied by supporting instances. Many attribute evaluation metrics have been proposed throughout the data mining and statistics community with two of the most popular metrics being the *Information Gain* metric (used in the C4.5 decision tree classification algorithm discussed in section 4.1.2) and the *Gini Index* (used in the Classification and Regression Trees (CART) algorithm [119]). The mathematical overview of these two metrics will now be presented. Each metric measures the impurity (heterogeneity) of the response variable at node T in relation to an attribute's ability to reduce this impurity. For the Information Gain metric presented earlier in section 4.1.2, this impurity reduction is achieved by maximizing the Information Gain equation, succinctly represented below (note: for a detailed overview of the Information Gain metric, please refer to section 4.1.2):

$$Information\ Gain(T) \quad = \quad (-\sum_{i=1}^{N} p(c_i) \cdot log_2 p(c_i)) - (\sum_{j=1}^{K} \frac{T_j}{T}(-\sum_{i=1}^{N} p(c_i|a_j) \cdot log_2 p(c_i|a_j)))[bits] \qquad (4.37)$$

Here,

- $p(c_i)$: represents the fraction of a response variable c_i in the data set T.

- $p(c_i|a_j)$: represents the fraction of a response variable c_i when conditioned on a particular attribute value a_j.
- N: represents the number of mutually exclusive response values within the data set (discrete case).
- $\frac{T_j}{T}$: represents the fraction of the data set that exists in the attribute partition a_j ($\sum_{j=1}^{K} \frac{T_j}{T}$=1).
- a_j : represents a unique value of the current test attribute A ($a_j \in$ A).
- K: represents the number of mutually exclusive attribute values of the current test attribute A.

The attribute that maximizes equation 4.37 is selected as the split attribute at each iteration until a purely homogeneous response variable exists in the data subset. The Gini Index presents a similar formulation as the Information Gain metric with the mathematical formulation as follows:

$$Gini\ Index(T) \quad = \quad (1 - \sum_{i=1}^{N} p(c_i)^2) - (\frac{T_L}{T}(1 - \sum_{i=1}^{N} p(c_i|a_L)^2) + \frac{T_R}{T}(1 - \sum_{i=1}^{N} p(c_i|a_R)^2))) \tag{4.38}$$

Here,

- $p(c_i)$: represents the fraction of a response variable c_i in the data set T.
- $p(c_i|a_L)$: represents the fraction of a response variable c_i when conditioned on the left attribute partition a_L.
- $p(c_i|a_R)$: represents the fraction of a response variable c_i when conditioned on the right attribute partition a_R.
- N: represents the number of mutually exclusive response values within the data set (discrete case).
- a_L : represents the left partition of the test attribute (binary attribute splits).
- $\frac{T_L}{T}$: represents the fraction of the data set that exists in the left attribute partition a_L ($\frac{T_L}{T}+\frac{T_R}{T}$=1).
- a_R : represents the right partition of the test attribute (binary attribute splits).
- $\frac{T_R}{T}$: represents the fraction of the data set that exists in the right attribute partition a_R ($\frac{T_L}{T}+\frac{T_R}{T}$=1).

In many systems design scenarios, engineers and design teams must make decisions about complex systems that will be evaluated on more than one objective. Consequently, existing single response data mining approaches may be insufficient in capturing next generation systems performance requirements.

4.5.2 Existing Multi-response Predictive Data Mining

The complexities of many real life design problems demand higher dimensions of performance evaluation in order to provide feasible candidate design solutions. The output from multi-response data mining models can then be set as targets in multi-objective engineering design models.

There have been several data mining approaches that have been proposed to handle multi-response variables. Zhang proposes a generalized entropy measure to handle multiple binary response variables [124]. This is achieved by maximizing the log likelihood function derived from an exponential distribution used to fit multiple binary responses. This approach however would severely limit design scenarios involving responses that have more than a binary output. Siciliano and Mola investigate multi-response variables that are not limited to binary cases and use a weighted Gini index as a measure of multi-response impurity that can be mathematically represented as[125]:

$$h(t) = (1 - \sum_{g=1}^{G}\sum_{i=1}^{N} w(g) \cdot p(c_{g,i})^2) - (\frac{T_L}{T}(1 - \sum_{g=1}^{G}\sum_{i=1}^{N} w(g) \cdot p(c_{g,i}|a_L)^2 + \frac{T_R}{T}(1 - \sum_{g=1}^{G}\sum_{i=1}^{N} w(g) \cdot p(c_{g,i}|a_R)^2)) \quad (4.39)$$

Here,

- p($c_{g,i}$:) represents the fraction of a multi-response variable (g) with given value (i) g∈G, i∈N.
- $p(c_{g,i}|a_L)$: represents the fraction of a multi-response variable (g) with given value (i) when conditioned on the left attribute partition a_L.
- $p(c_i|a_R)$: represents the fraction of a multi-response variable (g) with given value (i) when conditioned on the right attribute partition a_R.
- N: represents the number of mutually exclusive response values within the data set (discrete case).
- a_L : represents the left partition of the test attribute (binary attribute splits).
- $\frac{T_L}{T}$: represents the fraction of the data set that exists in the left attribute partition a_L ($\frac{T_L}{T}+\frac{T_R}{T}$=1).
- a_R : represents the right partition of the test attribute (binary attribute splits).
- $\frac{T_R}{T}$: represents the fraction of the data set that exists in the right attribute partition a_R ($\frac{T_L}{T}+\frac{T_R}{T}$=1).
- w(g): represents the assigned weight to multi-response variable (g).$\sum_{g=1}^{G}$w(g)=1 and w(g)≥0∀g=1,...,G.

One of the challenges with the above formulation is that the weights $w(g)$ can be subjective, creating biases in the attribute evaluation metric and subsequent decision tree model [126]. A related multi-response approach has been

proposed by Kim and Lee which extends the original information gain metric in equation 4.37 to include multiple responses [126]. This approach however may result in an exhaustive combination of multiple response values rendering the metric incapable of reducing the impurity of the multi-response data set. To illustrate this scenario, 3 response variables are given, each with 2 mutually exclusive values. That is, C1=$\{c_{1,1}, c_{1,2}\}$, C2=$\{c_{2,1}, c_{2,2}\}$, C3=$\{c_{3,1}, c_{3,2}\}$. There are a total of 2^3=8 possible combinations. Therefore if each of the response combinations is to be present in the data set, a minimum of 8 instances are needed to represent each unique combination. For a tree generation (data set partitioning) strategy to be feasible, the instance-response proportion must be high, otherwise a large tree may result with single instances existing at the final leaf.

In order to avoid the exhaustive multi-response combinations, an Apriori-like algorithm is first employed to the set of responses to determine the *frequently* occurring response combinations. Unlike the existing multi-response approaches discussed in the previous section, this enables subsets of response combinations to serve as the evaluation metric in the absence of a suitable number of complete response combinations. The multi-response combinations will satisfy the anti-monotone Apriori property: *if any length k pattern is not frequent in the database, its length (k+1) super-pattern can never be frequent* [127]. The example below will help illustrate the benefit of multi-response frequent pattern analysis as a means of generating a stable attribute performance metric. In Table 4.5, response C1 has a purely homogenous distribution with values $c_{1,1}$.

Table 4.5: Attribute Characterization based on Attribute Definition

Data Instance	**Response C1**	**Response C2**	**Response C3**
Instance 1	$c_{1,1}$	$c_{2,1}$	$c_{3,1}$
Instance 2	$c_{1,1}$	$c_{2,2}$	$c_{3,1}$
Instance 3	$c_{1,1}$	$c_{2,3}$	$c_{3,2}$
Instance 4	$c_{1,1}$	$c_{2,4}$	$c_{3,3}$
Instance 5	$c_{1,1}$	$c_{2,5}$	$c_{3,3}$

If this were a single response model, the tree would terminate at this data subset due to the purely homogenous distribution. In the multi-response scenario however, responses C2 and C3 must also be investigated. It can be observed that response C2 is the opposite of response C1 with a purely heterogenous distribution of values ranging form $c_{2,1}$ to $c_{2,5}$. If multi-responses C1 ,C2 and C3 were being evaluated simultaneously, this would yield 5 unique combinations, representing the most impure response distribution case. Instead, it is assumed that a minimum threshold for *multi-response frequency* is 2 instances within the data set. By employing the Apriori frequent pattern analysis on the multi-response cases, there are 2 response combinations left that satisfy the minimum threshold constraint. That is, combinations $\{c_{1,1}$ and $c_{3,1}\}$ and $\{c_{1,1}$ and $c_{3,3}\}$. Therefore, the attribute evaluation metric that is presented in the following section will judge an attribute's predictive power based on the frequent response combinations as determined by the Apriori algorithm, rather than an exhaustive combination of all possible response combinations.

Multi-response Attribute Evaluation Metric

The Generalized Association Measure (GAM) is proposed to handle design scenarios involving multi-response outputs. The GAM represents an attribute evaluation metric which is mathematically defined as[128].

$$GAM = \sqrt[k]{\frac{sup(\mathbf{X})^k}{n}(\frac{1}{sup(a_i)^k} + \frac{1}{sup(c_1)^k} ... + \frac{1}{sup(c_m)^k})} \tag{4.40}$$

Here,

- **X**: represents a set containing n events (a_i, and c_1...,c_m). a_i represents the value of the test attribute and c_1...,c_m (m $\geq$2) represent the set of multi-response values.
- n: represents the total number of events including the attribute value of interest (a_i, and c_1...,c_m).
- sup: represents the number of instances of an event(s) within the data set/subset.
- k: represents the exponent of the generalized mean. The given value of k enables the Generalized Association Measure to take the form of one of the null invariant measures such as: AllConf ($k \longrightarrow$ -∞), Coherence ($k = -1$), Cosine ($k = 0$), Kulc ($k = 1$) and MaxConf ($k \longrightarrow +\infty$). Each of these special cases of the Generalized Association Measures possess the *null-invariant* property which distinguishes them from many existing metrics. That is, their *interestingness* score is not influenced by irrelevant instances in the data set that do not contain the event combination of interest [128].

When investigating an attribute's predictive power through time, the stability of the multi-response attribute evaluation metric becomes critical. The following section discusses the stability of the Generalized Association Measure compared to traditional metrics such as the Information Gain.

Stability comparison of multi-response metrics

Table 4.6: Data set illustrating metric stability

Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Response
$a_{1,1}$	$a_{2,1}$	$a_{3,2}$	$a_{4,2}$	$a_{5,1}$	c_1
$a_{1,2}$	$a_{2,1}$	$a_{3,1}$	$a_{4,1}$	$a_{5,2}$	c_1

To gain a deeper understanding of the implications of these challenges, 3 different scenarios are presented with the results given in Table 4.7. For simplicity, the instances in Table 4.6 will serve as the basis for scenario 1. From Table 4.6, it can be observed that the attribute value $a_{2,1}$ is perfectly correlated with the response c_1 and would therefore

be unanimously considered as the attribute with the highest distinguishing power, regardless of the *interestingness* measure selected. It can be observe in the following 3 scenarios how the addition of irrelevant instances (null-data) can influence some *interestingness* measures.

- **Scenario 1:** The data set contains 2 instances as seen in Table 4.6.
- **Scenario 2:** 10 additional instances are added to scenario 1. For attribute 2, these 10 new instances assume the value ($a_{2,2}$) while the 10 additional response values assume the value (c_2), making $a_{2,2}$ and c_2, perfectly correlated.
- **Scenario 3:** Similar to scenario 2 but in this case, each of the 10 instances of both the attribute and response is unique however still perfectly correlated. That is, the 10 newly added instances of attribute 2 range from $a_{2,2}$ to $a_{2,11}$ and the 10 newly added response instances range from c_2 to c_{11}.

Table 4.7: Stability of Attribute Evaluation Measures

Evaluation Metric	Formulation	Bound	Scenario 1	Scenario 2	Scenario 3
Information Gain	$Entropy(Class) - Entropy(Class\|Attribute)$	$\{0,\infty\}$	0.00	0.650	3.418
GAM	$\sqrt[k]{\frac{sup(\mathbf{X})^k}{n}(\frac{1}{sup(a_i)^k} + \frac{1}{sup(c_1)^k} \ldots + \frac{1}{sup(c_m)^k})}$	$\{0,1\}$	1.00	1.00	1.00

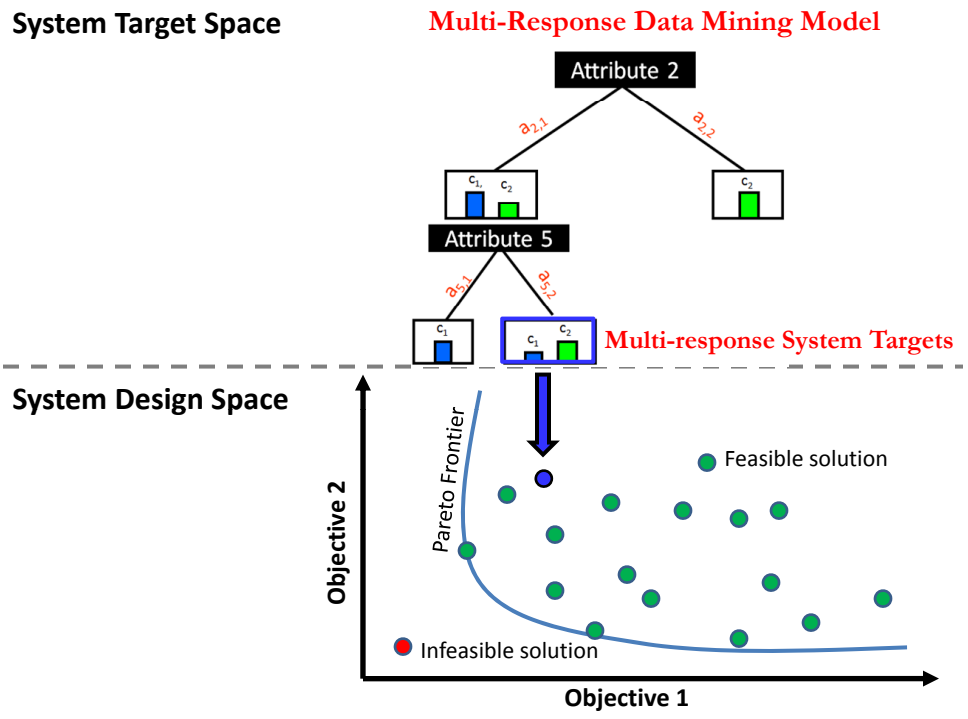


Figure 4.4: Linking Multi-response Data Mining with Systems Design

Although there exists perfect correlation between the attribute and response value in each of the scenarios, Table 4.7 reveals some of the inconsistencies that are inherent in the actual numeric interpretations of different attribute evaluation metrics. As can be seen from Table 4.7, the Generalized Association Measure (GAM) presents a more consistent numeric solution for perfectly correlated attribute-response relationships. This becomes critical when modeling the time series attribute behavior as time series modeling techniques may be sensitive to such variations in attribute correlation statistics.

4.5.3 Multi-Response Preference in Engineering Design

The complexity of many real life systems design problems demand higher dimensions of performance evaluation in order to provide more robust design solutions. Single system objectives such as cost minimization or environmental impact minimization are in many cases not mutually exclusive performance objectives and should therefore be investigated simultaneously during the design phase. The proposed Multi-Response Data Mining methodology can therefore be integrated into the design process and serve as targets to multi-objective, multi-level design models as seen in Figure 4.4. The attributes accompanying the multi-response model in Figure 4.4 can serve as constraints and parameters for the design variables involved in achieving the performance objectives in the multi-objective design space.

4.6 Comparative Analysis Between Demand Modeling Techniques

This section presents a comparative study of choice modeling and classification techniques that are currently being employed in the engineering design community to understand customer purchasing behavior. An in-depth comparison of two similar but distinctive techniques – the Discrete Choice Analysis (DCA) model and the C4.5 Decision Tree (DT) classification model – is performed, highlighting the strengths and limitations of each approach in relation to customer choice preferences modeling. A vehicle data set from a well established data repository is used to evaluate each model based on certain performance metrics: how the models differ in making predictions/classifications, computational complexity (challenges of model generation), ease of model interpretation and robustness of the model in regards to sensitivity analysis, and scale/size of data. The results reveal that both the Discrete Choice Analysis model and the C4.5 Decision Tree classification model can be used at different stages of product design and development to understand and model customer interests and choice behavior. However, the C4.5 Decision Tree may be better suited in predicting attribute relevance in relation to classifying choice patterns while the Discrete Choice Analysis model is better suited to quantify the choice share of each customer choice alternative.

Choice modeling through Discrete Choice Analysis (DCA) and classification through Decision Tree (DT) data mining modeling techniques are becoming well established approaches in the product design community to character-

ize customer choice behavior. The product design process requires many complex decisions that arise across multiple disciplines ranging from enterprise driven profit maximization to engineering product design optimization. In real life product design scenarios, many of these interactions are represented in the form of large-scale, heterogeneous data often containing great uncertainty through variations in customer preferences. These customer preference variations present a challenge to product development engineers trying to quantify and incorporate relevant product attributes into the final product design. In new product development, design engineers may want to predict the choice share of their brand (in relation to other brands) and understand how the absence or presence of particular attributes influence customer choice behavior. A quantifiable measure of attribute importance would enable both enterprise decision makers and design engineers better understand customer preference trends and product functionality demands as they relate to enterprise profit realization.

In the transportation research domain, several researchers have conducted comparative studies between the Discrete Choice Analysis models (in particular, the multinomial and nested logit models) and data mining classification models (in particular, the C4.5 Decision Tree classification model) [129, 130, 131, 132, 133]. The focus of these studies has been to identify choice patterns (i.e., modes of transportation) based primarily on an individual's behavioral and demographic information. The findings from the transportation mode studies reveal that the predictive capabilities of both the Discrete Choice Analysis models and the Decision Tree classification models are quite comparable [130, 131]. Although some distinguishing factors remain between the two techniques such as computation speeds and ease of interpretability, both techniques can be applied to modeling customers' transportation choices. In the transportation domain literature however, the choice set that customers are presented with is limited to a few alternatives and focuses primarily on socio-demographic attributes to predict customer choice behavior [129, 130, 131]. The size of the data sets used in these studies is also quite manageable when compared to the large scale data sets that have been used in data mining related product design problems [55].

Research Motivation:

In the product design research domain, no comparative study has been reported between Discrete Choice Analysis and data mining Decision Tree techniques. This work seeks to investigate the effectiveness of employing Discrete Choice Analysis and Decision Tree classification to specific product design scenarios involving a wide range of product design alternatives and heterogeneous customer preference data. As data acquisition and storage capabilities become less costly, there is a growing trend towards large scale data analysis and knowledge discovery. The increase in customer preference data may also lead to an increase in the number of attributes and product choice alternatives being investigated and may result in the *curse of dimensionality* problem, a situation that results when the performance of a model significantly diminishes as the dimension of the data set increases. This presents a tremendous design challenge when trying to identify significant attributes and choice alternatives within a high dimensional data set. It

is therefore necessary to investigate demand modeling techniques currently being employed in product design and development in order to understand their future potential in handling large scale customer preference data. Under this design paradigm, a comparison is made between the well established C4.5 Decision Tree (DT) classification technique and the Discrete Choice Analysis modeling (DCA) techniques that are so frequently used in the product design research community [21, 22, 23, 24].

The comparative study between the DCA technique and DT classification technique aims to address several key research questions:

1. **Research Question 1: Model Building Complexity**

   How much time do design engineers have to spend on customer choice data analysis and model construction?

2. **Research Question 2: Model Usage**

   How do the models differ in terms of the potential usage?

3. **Research Question 3: Quantifying Attribute Importance**

   How do design engineers quantify the relevance of different attributes (physical design attributes as well as socio-demographic attributes) in the design of next generation products?

4. **Research Question 4: Model Scalability**

   How easily can data sets and attribute dimensions be scaled and what effect does this have on the model integrity and complexity?

4.6.1 Vehicle Choice: A Comparative Case Study

The Experimental Data Set

A well established data repository of vehicle data developed by the University of California, Irvine is used to study the effectiveness of the Discrete Choice Analysis and the C4.5 Decision Tree classification techniques in product design [134]. The particular vehicle data set used in the comparative case study is based on real life data and has been successfully used by researchers in other fields such as the Computer Science and System Engineering to validate other unrelated research topics [135, 136, 137]. The selection of this data set for the comparative study was done primarily due to its relevance to product design related problems (ex. vehicle design) and enables the reader to independently investigate the findings of this work through the freely accessible University of California, Irvine data repository. Future research investigations aim to expand on the number of open source product development data sets used in the analysis in order to strengthen the theoretical findings of the demand modeling comparative study. The choice model

and decision tree estimated from this data set are intended to demonstrate the features of these techniques and do not necessarily represent optimal models for use in actual design.

Table 4.8: Physical and performance based attributes for the different vehicle types

Attribute Names	Attribute Values
aspirations	std, turbo
num-of-doors	four, two
num-of-cylinders	two, three, four, five, six, eight, twelve
horsepower	continuous from 48hp to 288hp
peak-rpm	continuous from 4150-6600
city-mpg	continuous from 13-49
highway-mpg	continuous from 16-54
price	continuous from 5118-45400
body-style	hardtop, wagon, sedan, hatchback, convertible

Table 4.9: Socio-demographic based attributes based on consumer identity

Attribute Names	Attribute Values
annual income	continuous from $25,000 to $120,000
age	continuous from 21-60
education level	high school, college, graduate
marital status	single, married
ethnicity	8 distinct categories
gender	male, female

Table 4.10: Class variable (response variable) which is the make of a particular vehicle

Attribute Names	Attribute Values
class	alpha romeo, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedez-benz, mercury, mitsubishi, nissan, peugot, plymouth, porche, renault, saab, subaru, toyota, volkswagen, volvo

The vehicle data set has been partitioned into two types of attribute information. First is the vehicle physical attribute information in Table 4.8 which includes performance and dimension specific attributes. In this multivariate data set, attributes can be numeric (having either discrete integer values or a continuous numeric range of values) or nominal (which do not contain an ordering of the attribute values such as the *color* of a vehicle or *name of a dealership*).

The demographic attributes in Table 4.9 make up the second part of the data set and include specific customer behavioral attributes such as *age, marital status, annual income*, etc. As with the vehicle physical attributes, demographic attributes can be either numeric or nominal attributes. The demographic attributes were not part of the original vehicle data set but were included in the analysis to be more consistent with the Discrete Choice Analysis model. Demographic attributes were simulated using MS Excel to reflect natural demographic correlations of customers based

on online sources investigated by the authors. For example, the simulated code accounted for higher likelihood that an individual with a higher education level would more likely have a higher annual income as compared to an individual with a lower education level.

The class variable presented in Table 4.10 can be thought of as the output given a specific attribute combination. In the vehicle data set used for the comparative study, the vehicle *make* is what is defined to be the class variable with class values ranging from Alfa-Romero to Volvo, all of which are nominal values. Therefore for a given combination of vehicle physical attributes and demographic attributes, a vehicle make is predicted.

Table 4.11 represents a snapshot of the data set wherein a given combination of attributes yields a preference in the class variable (make of the vehicle selected). Together, the physical vehicle attributes and the demographic attributes are used to quantify the response variable, *vehicle class*.

Table 4.11: Snapshot of vehicle data with both physical and demographic attributes, and the class variable (Make)

Aspiration	Number of doors	Number of Cylinders	Horse Power	Peak RPM	City MPG	Highway MPG	Vehicle Price	Body Style	Annual Income	Age	Education Level	Marital Status	Ethnicity	Gender	Make
stadard	2	4	288	4900	31	36	45400	3	46816	40	High Schoo	Single	8	female	subaru
stadard	4	4	262	5100	47	53	41315	3	66590	54	Graduate	Married	1	female	chevrolet
stadard	4	4	207	5000	30	31	40960	3	97504	45	Graduate	Single	2	male	mazda
stadard	2	4	207	4800	35	39	37028	3	90563	39	College	Married	5	male	toyota
stadard	2	4	207	5500	37	41	36880	3	25847	38	High Schoo	Single	6	female	mitsubishi

4.6.2 Results and Discussion

Product Design Research Questions

Research Question 1: Model Building Complexity

How much time do design engineers have to spend on customer choice data analysis and model construction?

This research question was addressed based on several key measures of *complexity*. First is the complexity of handing the raw data set and transforming it into an acceptable form for the Discrete Choice Analysis and C4.5 Decision Tree classification models. For the DCA model, dummy variables had to be created for each attribute where the ordering of attribute values was not desired. Fortunately this only had to be done for the body style vehicle attribute to avoid bias towards one type of vehicle body style (ex. sedan, hatchback, convertible, etc.) over another. In addition to this, the choice set of each customer has to be defined. In the event that the choice set is non uniform for all customers within the data set, this may present a great challenge in trying to develop the DCA model. Overall, the computational time needed to generate the discrete choice model using the Stata ® software package was approximately 3 seconds running on an Intel Pentium Duo 2.5 GHz Processor.

For the DT classification model, there was little preprocessing of the raw data set required, however the data set had to be in a comma separated value (CSV) file format or the ARFF file format to be compatible with the Weka 3.5

data mining package that was employed in this study [2]. The overall computational time needed to generate the data mining classification model was approximately 0.08 seconds running on an Intel Pentium Duo 2.5 GHz Processor.

Some factors that may have contributed to the significant disparity between the two demand modeling techniques may lie in their underlying approach to model development. The DCA model uses the *maximum likelihood* estimation to develop the parameters of the model which can be computationally expensive: especially if there exists noticeable correlations among the attributes. The C4.5 DT model on the other hand is an iterative induction technique that continues to partition the data set, hence making the *information gain ratio* calculation less computationally expensive with each iteration (partitioning of the data set into smaller subsets).

Research Question 2: Model Usage:

How do the models differ in terms of the potential usage?

To address this product design research question, the model prediction results of both the Discrete Choice Analysis and the C4.5 Decision Tree classification are analyzed for their respective usage. A primary question concerns what each model is actually predicting/classifying. The DCA model is a probabilistic model which uses a utility function to predict a given person's probability of choice of a given alternative as a function of both attributes of the product and attributes of the person. The DT model is a classifier which classifies a set of alternatives (i.e., classes) based on attribute values. The DT model determines attributes to be included in the tree based on the information gain, i.e., how effective is an attribute in differentiating the alternatives. These two techniques can be compared to determine the information each provides to the design process, although a direct comparison may not be straightforward. This is due to the different problem context that each model is used for. However, there have been successful comparisons of these two approaches in the transportation domain as discussed in the introduction section of this work [129, 130, 131, 132, 133].

The DCA model provides a model which considers the impact of both product and demographic attributes on the choice process. The model beta parameters are shown in Table 4.12. Such a model allows parametric studies to be conducted to determine the effects of changing product attributes, changing the target population, or both, on choice probability and hence choice share for a product of interest.

Because the model is parametric, it can easily be incorporated into an optimization framework to identify the product design attributes which maximize choice share of a given product, or a more advanced application can be used to identify product attributes which maximize enterprise level profit for a given target population as discussed again later in research question 4.

As discussed in the previous section, potential issues with the DCA model are that multicollinearity may inhibit the inclusion of all desired product or demographic attributes of interest in the model. Also, the MNL suffers from the Independence of Irrelevant Alternatives (IIA) property, which restricts the overall change in choice share due to

Table 4.12: Vehicle attribute importance based on beta (β) coefficients of the Discrete Choice Model

Vehicle Choice Model						
Conditional (fixed-e	logistic	regression		Number of obs	1751.00	
				LR chi2(70)	145.47	
				Prob > chi2	0.00	
Log likelihood = -359.88406				Pseudo R2	0.168	

Vehicle Attributes	**Coefficient**	**Std Error**	**z**	**P>|z|**	**95% Confidence Interval**	
vehicleprice	**-6.83**	2.43	-2.81	0.005	-11.59	-2.07
numberofdoors	-0.65	0.38	-1.74	0.081	-1.39	0.08
numberofcylinders	**-5.27**	1.85	-2.85	0.004	-8.90	-1.64
horsepower	**11.67**	2.32	5.03	0	7.12	16.21
peakrpm	0.02	0.72	0.02	0.982	-1.40	1.43
citympg	**3.25**	1.21	2.69	0.007	0.88	5.61
body-style_2	1.00	0.95	1.06	0.289	-0.85	2.86
body-style_3	0.63	0.79	0.8	0.423	-0.92	2.18
body-style_4	0.30	0.83	0.36	0.715	-1.33	1.94
body-style_5	-0.40	0.86	-0.46	0.643	-2.07	1.28

a change in a given attribute. This may result in unrealistic predictions of choice share changes for a given design change. Another issue with the DCA model is that because it is a parametric model, the underlying assumption is that the product and demographic attributes follow a standard functional relationship (i.e. linear, quadratic) with the underlying utility. This may impede its usage in situations in which utility does not follow a well defined functional relationship with product or demographic attributes.

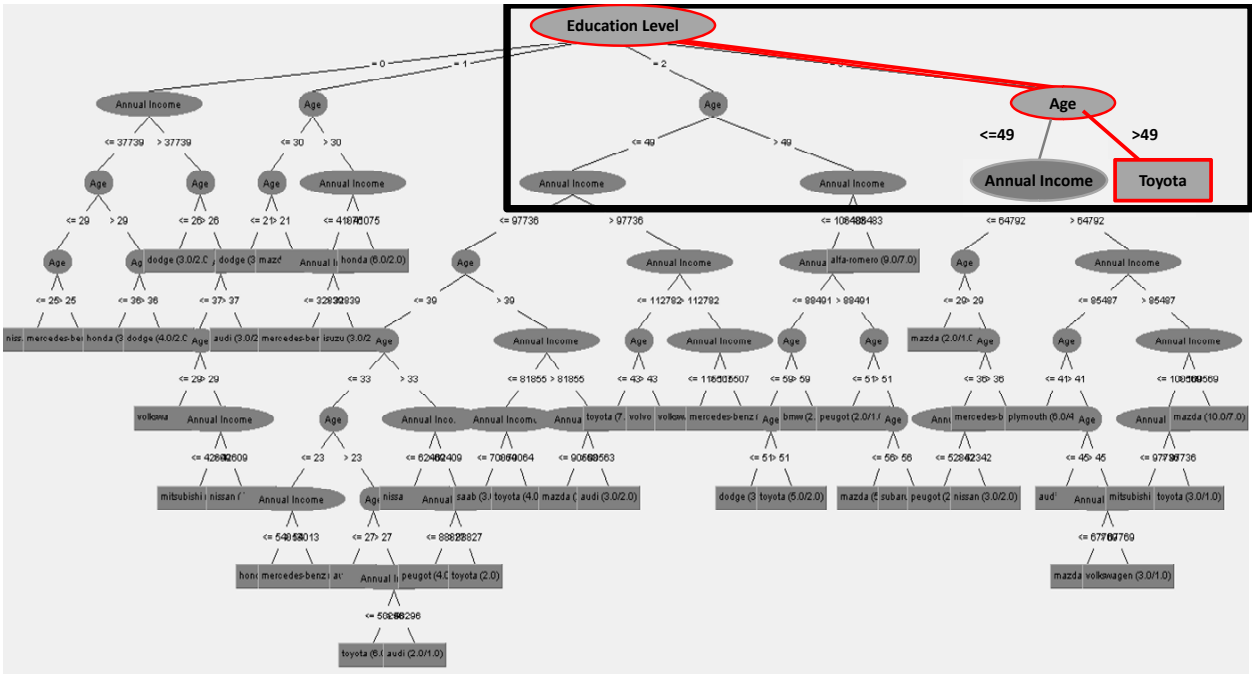


Figure 4.5: Visual Representation of C4.5 Decision Tree Model based on demographic attributes alone

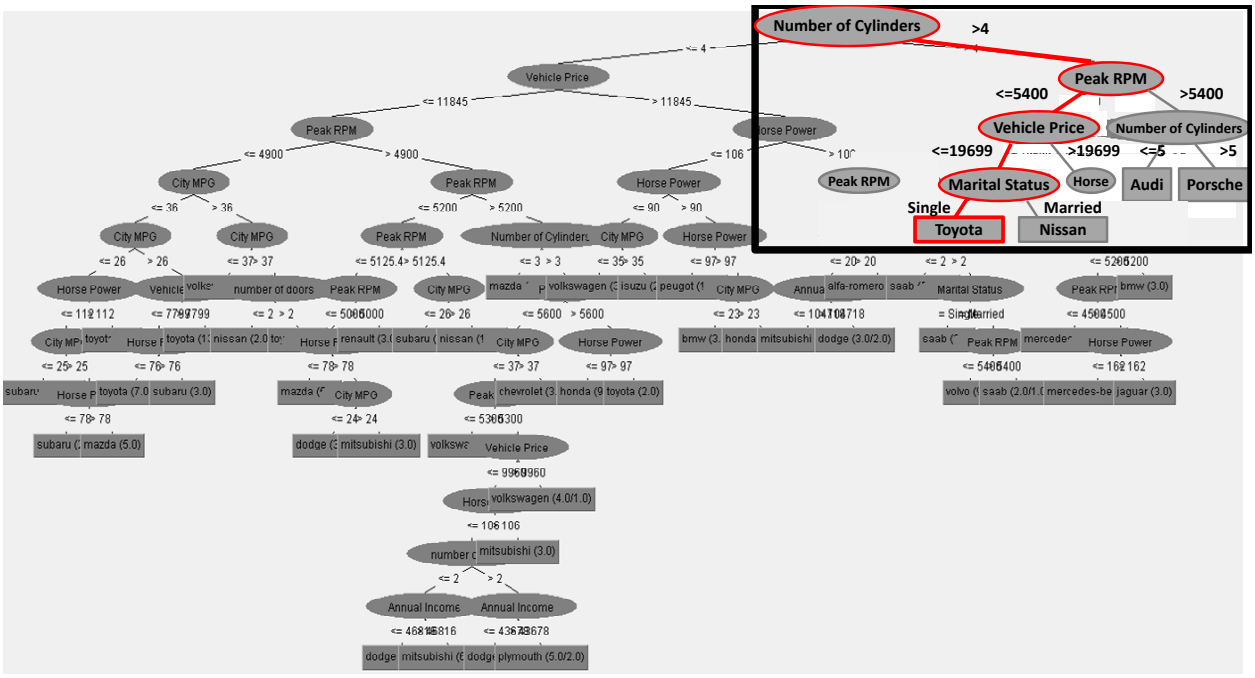


Figure 4.6: Visual Representation of C4.5 Decision Tree Model based on both product attributes and demographic attributes

The C4.5 Decision Tree shown in Figure 4.5 can be used to classify product choices as a function of demographic attributes and in Figure 4.6 can be used to classify both demographic and product attributes together from the data mining classification perspective. For example Figure 4.5 reveals that the most significant demographic that influences vehicle classification is the *Education Level* that an individual attains. Furthermore, by traversing down the Decision Tree path in Figure 4.5 (indicated in red), a classification rule is reached that states: If *Education Level=Graduate* **and** *Age>49===> Vehicle Class= Toyota*. For design engineers designing the next generation of vehicles, this vital piece of information would enable more customized design features for this specific demographic group. However there are several noticeable distinctions in the model predictions. For example in Figure 4.6, the most important attribute in differentiating, or classifying products, is number of cylinders, followed by vehicle price and peak RPM. Such attributes may or may not be important in the choice process, (as will be described in the next subsection) but do provide information to design engineers. Therefore knowledge that the vehicles can be differentiated by peak RPM can lead to a benchmarking study to understand why there is much variation in this attribute, leading to a better understanding of competitive designs. Although peak RPM may not be a significant choice attribute, it may be found that peak RPM is correlated with a performance that is important in the choice process through a technical analysis. Also, decision trees do not require a smooth functional relationship between attributes and choice. As shown in the Figures 4.5 and 4.6, the decision tree provides a more visual understanding of how choices are classified, which can be useful for designers who want to better understand how different segments of the population select different products.

The C4.5 Decision Tree built with **S** can be compared to a parametric Multinomial Logit Regression (MNR) model built using only **S** to predict choices. Figure 4.7 gives the histogram results of both the MNR and the C4.5 Decision Tree. For this given data set of only demographic attributes, the MNR and C4.5 Decision Tree are quite comparable on most of the vehicle choice predictions, except the Toyota brand which the MNR over predicts and the C4.5 under predicts. One reason for this disparity may be due to the fact that the Toyota brand is the most frequently occurring brand within the data set, and each of the models compensates for this brand bias differently. The C4.5 DT tends to over predict choice alternative more frequently than the DCA. However when the DCA over predicts, it does so by an extremely significant margin (as is seen with the Toyota brand). This provides tremendous insight as to what product development scenario each model may be better suited for. For example if consistent minor miscalculations in a product create significant backlash in the customer market segment, then product development engineers may opt to employ the DCA model. However if the consequences of having a large disparity between customer wants and engineering capabilities is high, then product development engineers may decide to employ the DT model instead which seemed to not over predict by great margins. Similar results (i.e., comparable prediction result) are also observed in Figure 4.8 when both demographic and product attribute data are considered.

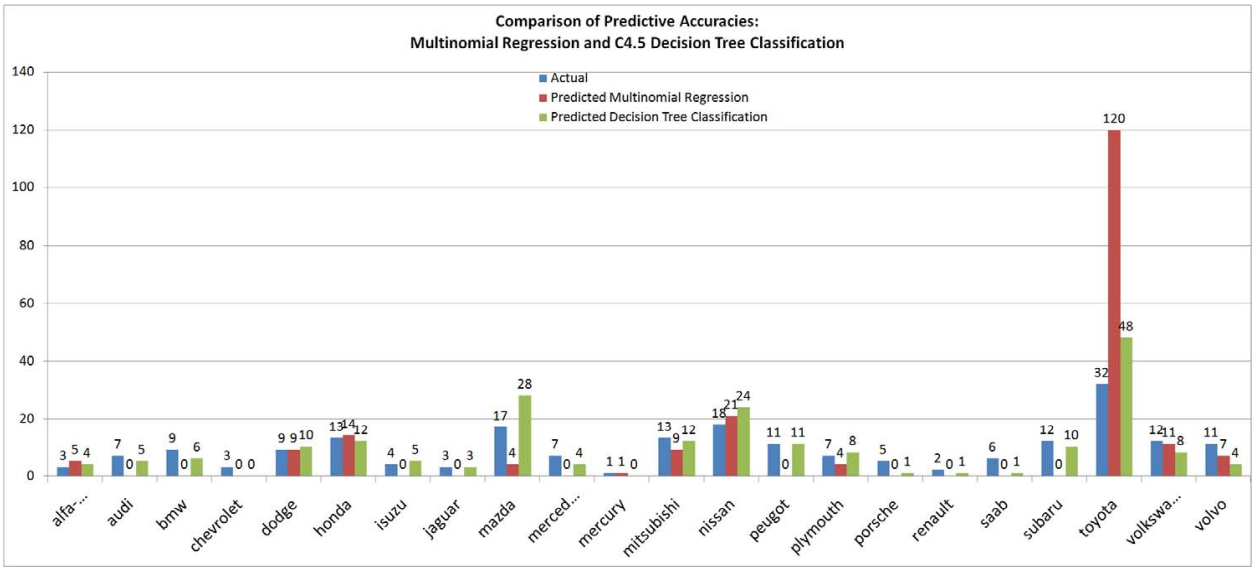


Figure 4.7: Histogram of prediction results of the MNR and C4.5 DT models based on demographic attributes alone

Research Question 3: Quantify Attribute Importance

How do design engineers quantify the relevance of different attributes (physical design attributes as well as socio-demographic attributes) in the design of next generation products?

Both Discrete Choice Analysis and the C4.5 Decision Tree classification employ different techniques to quantify attribute importance. In regards to the product specific attributes (which in the case study include attributes such

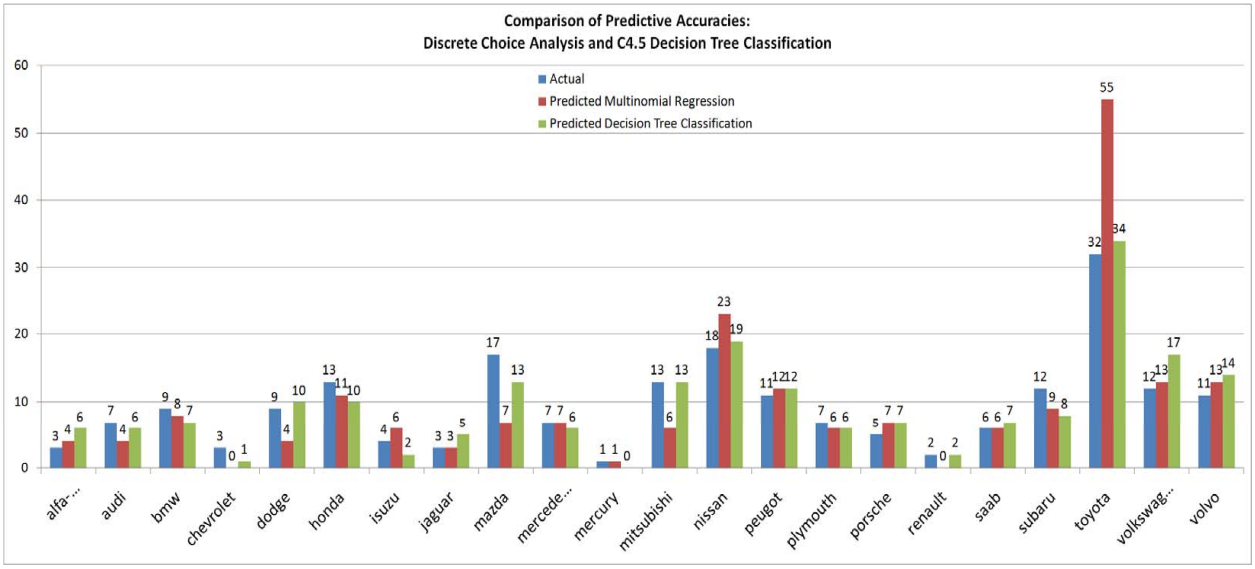


Figure 4.8: Histogram of prediction results of the MNR and C4.5 DT models based on demographic and product attributes

as *horse power, mpg, number of doors, etc.*), the DCA model uses the beta coefficients β to represent the level of significance a particular product attribute has in determining a customer's utility for a particular vehicle brand.

In the case study, since it is assumed that all vehicle choice alternatives have the same attributes (although they may have varying degrees of attribute values), the beta coefficient β for a given physical vehicle attribute is constant across all vehicle alternatives. Similar coefficients are calculated for the socio-demographic information, and a utility function represents significance of these attributes in predicting customer choice behavior of vehicle alternatives.

The C4.5 Decision Tree classification employs a different technique to quantify attribute relevance in product design and development. Based on the *gain ratio* criterion defined in equation (4.7), the C4.5 Decision Tree algorithm tests each attribute and iteratively partitions the data set based on the attribute that maximizes this *gain ratio*. The C4.5 Decision Tree structures presented in Figures 4.5 and 4.6 give a visual representation of attribute relevance to vehicle choice alternatives. The closer the attribute is to the root of the tree (top most point of the tree in Figure 4.5), the greater the *gain ratio* calculated for this attribute and hence the greater the attribute relevance to vehicle choice classifications.

Several attributes in Figure 4.5 appear more than once within the tree due to the fact that continuous attributes are discretized during the iterative partitioning of the data set and therefore different bounds for a particular attribute may exist for different branches within the tree.

An interesting comparison can be made between the decision tree built using product and demographic attributes and the choice model. As noted in the previous sub-section, product and demographic attributes important in the decision tree are those which are most important in terms of differentiating, and hence classifying, vehicles. Product

and demographic attributes important in the choice model are those which are important in terms of describing why people choose different vehicles. It is possible that attributes important in terms of classifying vehicles, such as peak RPM, are not important in the choice process and vice versa. Table 4.12 reveals that *Horsepower* is the most important choice attribute, whereas *number of cylinders* provided has the highest gain ratio and can be interpreted as being the most important attribute for classifying vehicles. While there are differences among the rankings of importance of attributes between the two methods, there is also much agreement. *Price*, *horsepower*, *city MPG*, and *number of cylinders* are important to both classifying vehicles as well as describing the choice process. This information is important to designers as it shows that vehicles do differ significantly on these attributes and they are important to the choice process. On the other hand, Peak RPM has high relative importance in the decision tree but is insignificant in the choice model. This tells the designer that peak RPM may be an attribute to study during design from a technical standpoint since it varies quite a bit among vehicles, but that it is not important in the choice process.

Both the Discrete Choice Analysis and the C4.5 Decision Tree classification techniques provide researchers and design engineers quantifiable measure of relative attribute importance in predicting product choice alternatives. Although the Discrete Choice Analysis gives more of a quantitative measure (β coefficients) while the C4.5 Decision Tree gives more of a qualitative measure (the position of the attribute within the tree structure) in the attribute importance context. However in the attribute classification process, the splitting of a quantitative attribute (for example horse power) in the decision tree can also give a quantitative measure of that specific attribute level.

In terms of model sensitivity, the structure of the C4.5 Decision Tree model in Figure 4.6 can be analyzed in detail. Traversing down the tree as indicated by the red path, that is, citation{*Number of Cylinders = 4, Peak RPM = 5400, Vehicle Price = $19699, Marital Status=Single*}, would result in a classification of *Class=Toyota*. Therefore for a design engineer trying to understand the effects of changes in product attributes in relation to customer preference, the decision tree rule stated above would indicate that as long as the peak RPM was less than 5400 (holding all of the other attribute values constant), the resulting classification would be the preference of the *Toyota* brand. As an enterprise decision maker of Toyota trying to maximize revenue, one would be tempted to decrease the peak RPM of the vehicle (while holding all other attribute levels constant, hereby possibly reducing the manufacturing costs of the vehicle). This may be theoretically sound based on the formulation of the C4.5 algorithm itself, however in reality, reducing the peak RPM of the vehicle beyond a certain lower bound may cause customers to forgo this vehicle brand altogether (e.g. if the vehicle had a peak RPM below all other customer choice alternatives).

The Discrete Choice Analysis technique approaches sensitivity analysis of attributes in a slightly different manner. Where the C4.5 Decision Tree generates hard decision rules, the Discrete Choice Analysis model in comparison updates the probability of choosing a particular brand of vehicles by once again generating a utility function based on updated attribute values.

Research Question 4: Model Scalability

How easily can data sets and attribute dimensions be scaled and what effect does this have on the model integrity and complexity?

The term *model scalability* used in this work relates to the ease at which the dimensions of the attribute space or the size of the data can be increased and does not refer to scaling factors that may be used as parameters in different demand modeling techniques. One of the major challenges of employing the Discrete Choice Analysis model in this work was the issue of multicollinearity when attempting to minimize the Maximum Likelihood Estimate (MLE). This is a frequent challenge often occurring in high dimensional data where correlations among attributes may exist more frequently. Multicollinearity, which is caused by high correlation among the product or customer attributes assumed to be independent, can cause unstable MLEs and inaccurate variances which may adversely affect the quality of the generated model [138]. To minimize the bias that may result due to the multicollinearity issues experienced by the Discrete Choice Analysis model, the attribute dimensions of the original vehicle data set were reduced to a more manageable size. There are several proposed techniques to minimizing the effects of multicollinearity which include the aggregation of highly correlated attributes into a newly defined attribute. One of the drawbacks of such an approach is the ambiguity that results in trying to determine which attribute influences the class variable.

Like many classification based techniques in data mining, one of the fundamental assumptions of the C4.5 Decision Tree approach is that attributes within the data set are independent. This assumption enables the C4.5 algorithm to sequentially test each attribute and determine its significance as it relates only to the class variable, overlooking the interaction effects that may exist among attributes. Although this assumption may seem nave to some, vast empirical data within the Data Mining community have validated the predictive accuracies of the C4.5 and other similar classification based approaches on their performance and predicative accuracy as it relates to traditional statistical methods that do take into account higher order interactions amongst attributes [139, 140]. Therefore, the C4.5 Decision Tree model is less affected by model scalability (in terms of high dimensionality attribute space) compared to the DCA model and may be more suitable for product development efforts involving high dimensional customer attribute data.

.

Relation to Transportation Domain Research

In comparison to the research findings in the transportation domain, issues of multicollinearity were not reported, one reason perhaps being the lower dimensionality attribute data used in that study in addition to the smaller choice alternative set investigated (5 choice alternatives in the transportation domain research, compared to 22 choice alternative is the product development study presented in this work). In addition to this difference, the transportation domain research focused only on customer demographic attributes (age, gender, household income, etc.) while the product development comparative study presented in this work included both customer demographic attributes (age, gender,

Table 4.13: Summary of the DCA and C4.5 DT techniques

Model	Model Generation Complexity	Attribute Dimensionality Limitations	Ease of Model Construction	Interpretability	Data Size
DCA	Straightforward and Efficient	Co-linearity issues in calculating MLE	Moderate	Utility Function	Small to medium
DT	Straightforward and Efficient	Requires entire data set to fit into main memory	Moderate	Decision Tree model	Medium to large

household income, etc.) as well as physical/performance product attributes (vehicle horsepower, MPG, vehicle body type, etc). The inclusion of physical product attributes in this work enhanced the understanding of the difference between the DCA and DT models in their predictive capabilities and model interpretation.

4.6.3 Summary and Conclusion

The overall research findings from the comparative study of the Discrete Choice Analysis (DCA) and C4.5 Decision Tree (DT) Classification approaches are summarized in Table 4.13 above. Both of the techniques are quite efficient in model generation (i.e., computation time). In terms of attribute dimensionality, several challenges due were encountered due to multicollinearity in the DCA model generation while the DT is constrained to the memory capabilities. Both methods are quite straightforward to implement.

The DCA models used in this work were generated using the Stata§package while the DT model was generated using Weka 3.5.1. The DCA generates a predictive model based on a given utility function that can be easily interpreted by design engineers and embedded in an optimization model. DT however presents a visual model that is mainly classification and qualitative in nature. The data set used in this case study can be considered small to medium scale (205 instances with 10 attributes and one classifier variable). For large scale data, DCA may experience some challenges with multicollinearity as discussed in section 5. The comparative study will be expanded to include a more diverse set of product development data sets (stated VS revealed preference data, large VS small data sets, high dimensional VS low dimensional attribute space, etc.) so as to present a more robust and generalized summary of the relevance of DCA and DT in product development.

In summary, two well established choice modeling techniques were presented in order to investigate their ability to quantify and classify customer choice behavior as it relates to product design and development. Real life vehicle choice data (revealed preference data) was used to compare the two techniques and present recommendations relating to product design. Each approach has its strengths and weaknesses ranging from computational complexity to model interpretability. Overall, both the Discrete Choice Analysis and the C4.5 Decision Tree can be used to model customer choice behavior in product design. However, the C4.5 Decision Tree may be better suited in predicting attribute

relevance in relation to classifying choice patterns while the Discrete Choice Analysis model is better suited to quantify the choice share of each customer choice alternative. Because the decision tree provides a visual representation of the classification process, it provides a useful tool for both quantitatively and qualitatively understanding how customers of different demographic attributes select vehicles and how products are differentiated. The choice model does not provide a visual representation, but provides an understanding of how both product and customer attributes influence the choice process. It is hoped that the findings in this work will serve as a guide to enterprise decision makers and engineers investigating efficient approaches to demand modeling and product design and development.

Chapter 5

Trend Mining for Predictive Product Design

5.1 Trend Mining Methodology

The ability to model emerging trends has broad applicability in product development, ranging from researching and developing new product technologies to quantifying changes in consumer preferences in highly volatile markets. Traditional demand modeling techniques frequently employed in the product design community typically generate predictive models using data from a single snapshot in time (usually the most currently available data set) and hence may not reflect the evolving nature of product trends. The absence of a temporal demand model for product design presents a challenge to design engineers trying to determine the relevant product attributes to include/exclude in the next generation of products.

The *Preference Trend Mining* (PTM) algorithm that is proposed in this chapter aims to address some fundamental challenges of current demand modeling techniques being employed in the product design community. The first contribution is a multistage predictive modeling approach that captures changes in consumer preferences (as they relate to product design) over time, hereby enabling design engineers to anticipate next generation product features before they become mainstream/unimportant. Because consumer preferences may exhibit monotonically increasing or decreasing, seasonal or unobservable trends, a statistical trend detection technique is employed to help detect time series attribute patterns. A time series exponential smoothing technique is then used to forecast future attribute trend patterns and generate a demand model that reflects emerging product preferences over time.

The second contribution of this chapter is a novel classification scheme for attributes that have low predictive power and hence may be omitted from a predictive model. Such attributes can be classified as either *standard*, *nonstandard* or *obsolete* with the appropriate classification given, based on the time series entropy values that an attribute exhibits. By modeling attribute irrelevance, design engineers can determine when to retire certain product features (deemed obsolete) or incorporate others into the actual product architecture (standard) while developing modules for those attributes exhibiting inconsistent patterns throughout time (nonstandard). Several time series data sets from the UC Irvine machine learning repository are used to validate the proposed Preference Trend Mining model and compare it to traditional demand modeling techniques for predictive accuracy and ease of model generation.

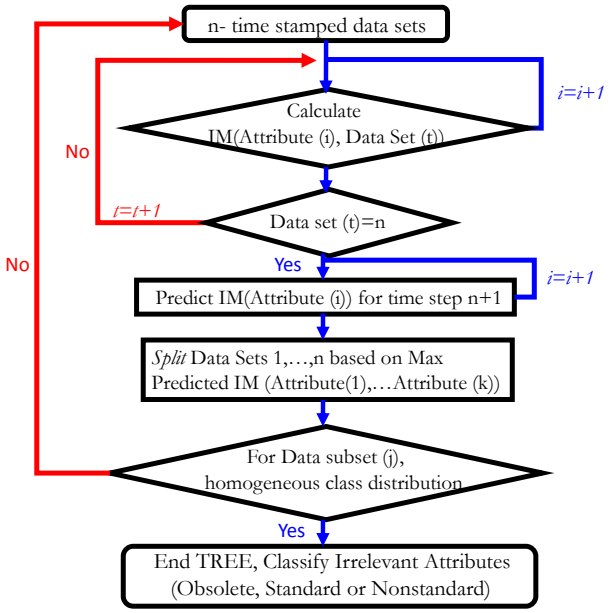


Figure 5.1: Overall Flow of Preference Trend Mining Methodology

Limitations of Current Demand Modeling Techniques

A recent comparative study in the product design community between the Discrete Choice Analysis and Decision Tree Classification models reveals that both techniques are quite comparable in terms of model generation and predictive accuracy. However the Decision Tree Classification model was found to be better suited for large scale data analysis due to multicollinearity issues reported while employing DCA for high dimensional data [141]. Nevertheless both demand modeling techniques are limited in their ability to characterize evolving product preference trends in the market space due to the static nature of the models. Because the input of each model typically represents an instant in time, design engineers are faced with the challenge of anticipating shifts in product preferences based on personal experience, rather than quantitative customer feedback. To overcome these challenges the Preference Trend Mining algorithm is introduced in the following section.

5.2 Methodology

Figure 5.1 presents the overall flow of the Preference Trend Mining algorithm, starting with the acquisition of *n* time-stamped data sets. For each time step, the *Interestingness* Measure (IM) is calculated for each attribute. There have been many proposed measures for evaluating attribute *interestingness* (relevance) such as the *information gain* metric, *gini* index, *Cosine* measure, *Support* measure, *Confidence* measure, to name but a few [128, 142]. In this work, the definition of attribute *interestingness* is limited to an attribute's ability to reduce the non-homogeneity of the class

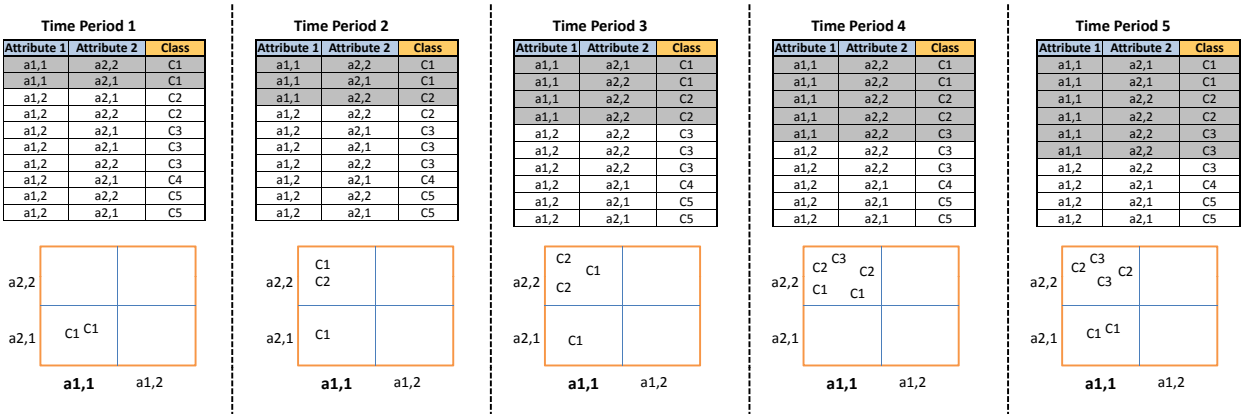


Figure 5.2: Attribute-class distributions over time (attribute a1,1 is highlighted although both attribute patterns change over time)

variable. In section 5.2.2, the inconsistencies that exist among different definitions of *relevance* are highlighted while proposing an approach to mitigate these inconsistencies by evaluating attribute *interestingness* through time. That is, an attribute that is truly relevant, will have consistently high relevance scores throughout time and vice versa.

For each time step in Figure 5.2, the IM for each attribute is calculated and a seasonal time series predictive model is then employed to forecast the trend patterns (monotonically increasing, decreasing or seasonal trend patterns) for each attribute. The attribute with the highest predicted IM is selected as the split attribute for the future (unseen) time period and all time stamped data sets are partitioned based on the unique values of this attribute. The process continues until a homogenous class value exists in the model. The flow diagram in Figure 5.1 ends with the classification of attributes (as either obsolete, standard and nonstandard) that are omitted from the resulting model.

The following sections of the paper will expound on the steps of the flow diagram in Figure 5.1.

5.2.1 Discovering Emerging Trends For Product Design

Trends within a data set can be characterized as monotonically increasing or decreasing, seasonal (where data exhibits some type of cyclical behavior) or both. There may also be instances where the time series data set exhibits no type of discernable pattern suitable for statistical modeling. In the context of product design, each of these preference trend scenarios are considered in the proposed methodology. The time series data set represented in Figure 5.2 will be used to illustrate the notion of attribute trends within a raw data set. Figure 5.2 comprises of 5 time periods. Attribute 1 comprises of two unique values $\{a_{1,1}, a_{1,2}\}$ and similarly for Attribute 2 $\{a_{2,1}, a_{2,2}\}$. The last column in Figure 5.2 represents the class (dependent) variable which has 5 mutually exclusive outcomes $\{c_1,c_2,c_3,c_4,c_5\}$. As can be observed from time periods t_1 to t_5, the number of instances of Attribute 1's value $a_{1,1}$ increases from 2 at time period t_1 to 6 at time period t_5. Looking closer at the square graphs in Figure 5.2, it can be observed that at time period t_1, although Attribute 1's $a_{1,1}$ value only has a total count of 2, it represents a homogenous distribution of class value c_1

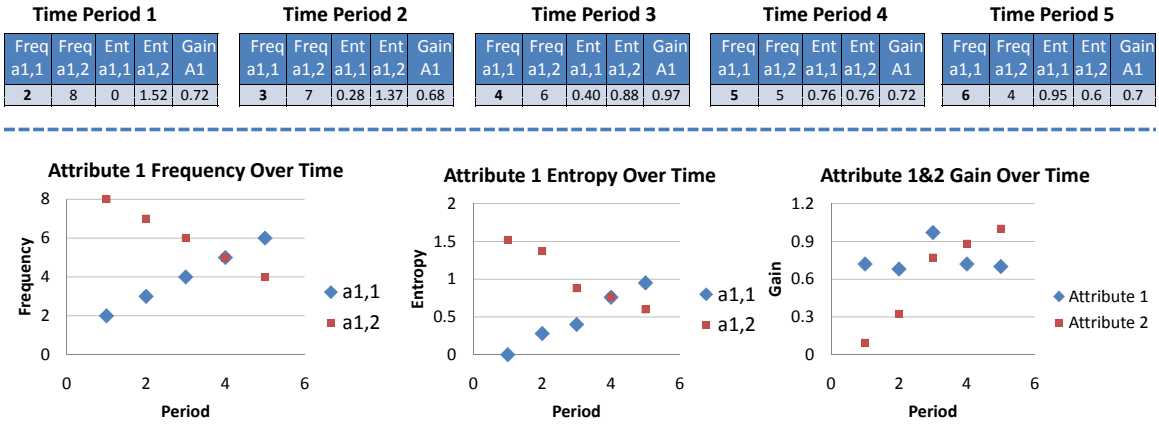


Figure 5.3: Characterizing Attribute Preference Trend Over Time

(lower left quadrant in time period t_1). Moving through time to time step t_5, it can be observed that the same attribute value $a_{1,1}$ has a count of 6 but with a non-homogeneous distribution of the class variable (the lower left quadrant in time series t_5 has a mixture of c_1, c_2 and c_3). The change in the predictive power of each attribute can be quantified by calculating the attribute *Interestingness* Measure (IM) over time which in this case is the *Gain Ratio*. Figure 5.3 presents a visual representation of each attribute *Gain Ratio* over time. In Figure 5.3, although Attribute 1 starts out with a higher *Gain Ratio* (predictive power) than Attribute 2, by time period 4, Attribute 2 has over taken Attribute 1 in *relevance* to the class variable. If a predictive model had been generated at time period 3, the emerging preference trend of Attribute 2 would not have been realized. To overcome these challenges, the Holt Winters exponential smoothing technique is employed which uses a weighted averaging technique, taking into account the local level, the trend, and the seasonal components of the time series data [143, 144].

Holt-Winters Exponential Smoothing

The Holt-Winters Exponential Smoothing is a non-parametric model that can be used to forecast each attribute's predictive power for the kth step ahead so that emerging preference trends can be anticipated in the market space. Non parametric statistical tests may be preferred in machine learning scenarios due to the relaxation of the normality assumption that many parametric statistical trend tests require [145]. Since no prior knowledge of the distribution of the incoming data is assumed, a relaxation of the data normality constraint is preferred. The (k) step-ahead forecasting model is defined as:

$$\widehat{y}_t(k) = L_t + kT_t + I_{t-s+k} \tag{5.1}$$

where

Level L_t (the level component):

$$L_t = \alpha(y_t - I_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{5.2}$$

Trend T_t (the slope component):

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1} \tag{5.3}$$

Season I_t (the seasonal component):

$$I_t = \delta(y_t - L_t) + (1-\delta)I_{t-s} \tag{5.4}$$

Here,

- y_t: Represents the data point at the most recent time period (t).
- $\widehat{y_t}(k)$: Represents the k^{th} time step ahead forecasted value beyond y_t (I.e.,$\widehat{y_t}(k)$= y_{t+k}).
- s: Represents the frequency of the seasonality (monthly, quarterly, yearly, etc.).

The smoothing parameters α,γ,δ are in the range $\{0,1\}$ and are estimated by minimizing the sum of squared errors for one time step ahead [143, 144].

Several well established statistical techniques (both parametric and nonparametric) exist for modeling time series data including the Seasonal-Trend decomposition procedure based on Loess regression (STL), variations of the Box-Jenkins models which include the AutoRegressive Moving Average (ARMA) and AutoRegressive Integrated Moving Average (ARIMA), to name but a few [146, 147]. Research studies on the predictive accuracies of these models reveal no conclusive evidence to suggest one model being superior for all data structures [147].

Based on the results in Figure 5.3, it can be observed that Attribute 2 would be selected as the relevant attribute in time period 6 (since at each iteration, the attribute with the highest *Gain Ratio* is always selected). Under the *Gain Ratio* definition of attribute relevance, Attribute 1 would now be considered *irrelevant* at iteration 1 of the decision tree induction algorithm. Based on the irrelevance characterizations presented in section 5.2.2, Attribute 1 could either be an *obsolete attribute*, a *nonstandard attribute* or a *standard attribute*. In order to determine the assignment of Attribute 1, the temporal behavior of each mutually exclusive value of Attribute 1 ($a_{1,1}$ and $a_{1,2}$) needs to be determined. The following section details the proposed attribute quantification methodology.

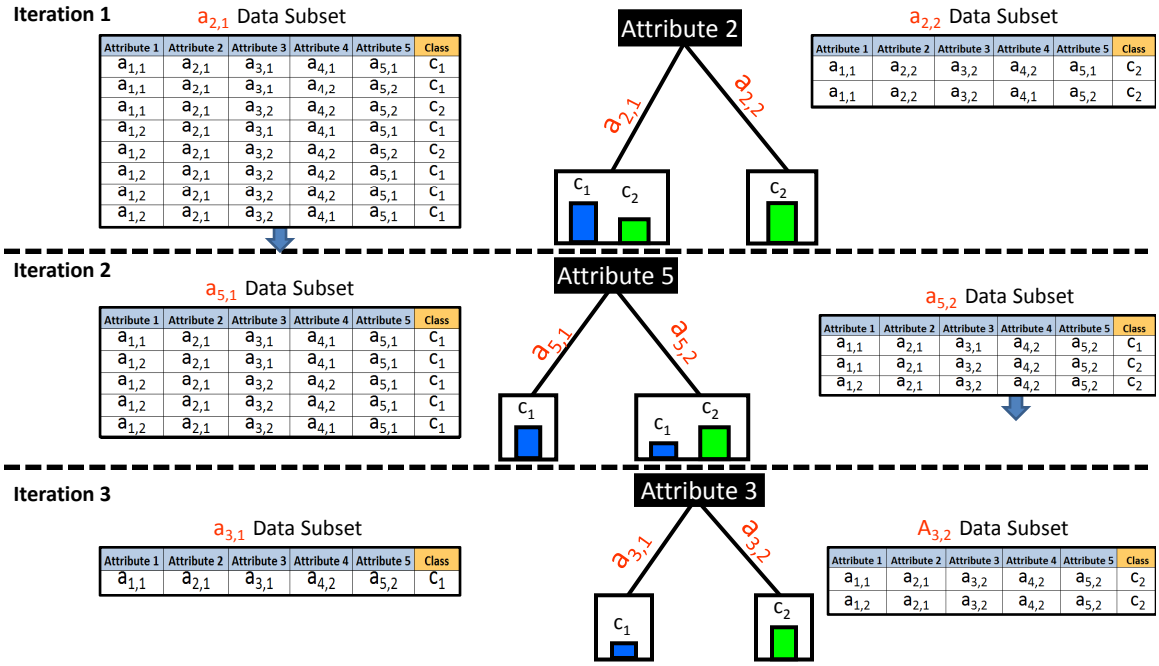


Figure 5.4: Example Decision Tree Result for Product Design

5.2.2 Quantifying Attribute Relevance

One of the major challenges in predictive model generation is understanding the design implications of the resulting model in terms of attribute *relevance* or *irrelevance*. To understand some of the challenges that arise in demand models, the following example is presented.

A set of attributes $\{A_1,..., A_5\}$ is defined, each with a set of mutually exclusive outcomes $a_{i,j}$, where i corresponds to the specific attribute A_i and j corresponds to the attribute value. For simplicity, it is assumed that $j = 2$ for all attributes. A *class* variable is also defined that is conditionally dependent on one or several of the defined attributes. The class variable is also binary with values $\{c_1 , c_2 \}$.

Figure 5.4 is a visual representation of a resulting data mining decision tree structure employing the *Gain Ratio* metric described in section 4.1.2. The following decision rules can be obtained by traversing down each unique path of the tree in Figure 5.4.

1. IF A_2=$a_{2,1}$ AND A_5=$a_{5,1}$ THEN Class=c_1
2. IF A_2=$a_{2,1}$ AND A_5=$a_{5,2}$ AND A_3=$a_{3,1}$ THEN Class=c_1
3. IF A_2=$a_{2,1}$ AND A_5=$a_{5,2}$ AND A_3=$a_{3,2}$ THEN Class=c_2
4. IF A_2=$a_{2,2}$ THEN Class=c_2

Looking at the four decision rules above, it can be observed that attributes A_1 and A_4 are not part of the model. Some immediate questions arise based on these findings:

1. What does the absence of attributes A_1 and A_4 tell design engineers about their *relevance* to future product designs?

2. How long into the future will the current decision rules be (i.e., maintain high predictive capability) *valid*?

3. Are there any emerging attribute trends that are not represented by the decision tree that may be useful to design engineers?

Table 5.1: Attribute Characterization based on Attribute Definition

Attribute	**D** 1	**D** 2	**D** 3	**D** 4	**D** 5
Attribute 1		x		x	
Attribute 2	x	x		x	
Attribute 3		x		x	x
Attribute 4		x		x	
Attribute 5	x	x		x	x

Before addressing the research questions regarding *attribute relevance/ irrelevance*, several well established definitions of attribute relevance that exist in the literature are introduced [148, 149].

- **Definition 1**: *An attribute A_i is said to be relevant to a* concept *(decision rule) C if A_i appears in every Boolean formula that represents C and irrelevant otherwise.*

- **Definition 2**: *A_i is relevant iff there exists some attribute value a_{ij} and class value c_i for which $p(A_i{=}a_{ij}){>}0$ such that $p(Class{=}c_i|A_i{=}a_{ij}){\neq}p(Class{=}c_i)$.*

- **Definition 3**: *A_i is relevant if each unique value varies systematically with category (class) membership*

- **Definition 4**: *A_i is relevant iff there exists some a_{ij}, c_i, and s_i for which $p(A_i{=}a_{ij}){>}0$ such that $p(Class{=}c_i,S_i{=}s_i|A_i{=}a_{ij}){\neq}p(Class{=}c_i,S_i{=}s_i)$* , where S_i represents the set of all attributes not including A_i.

- **Definition 5** *A_i is* **strongly relevant** *iff there exists some a_{ij}, c_i and s_i for which $p(A_i{=}a_{ij},S_i{=}s_i){>}0$ such that $p(Class{=}c_i|A_i{=}a_{ij},S_i{=}s_i) {\neq}p(Class{=}c_i \mid S_i{=}s_i)$.*

Based on the results from Table 5.1, there exists the possibility that an attribute evaluation metric may omit relevant attributes in the model due to inconsistencies in how attribute relevance is defined [148]. For design engineers, omitting a key attribute due to an irrelevance characterization could mean the subsequent failure of a product as customer needs may not be fully captured. These inconsistencies in attribute characterization are minimized by looking at the problem from a time series perspective. That is, attributes that are truly relevant to a product design should consistently show up in the predictive models through many time steps and attributes that are indeed irrelevant to a product design would remain absent in the predictive model over time.

The next section relates the concepts of attribute relevance to product design where the definition of attribute relevance/irrelevance is expanded to aid design engineers determine when to include or exclude certain attributes for next generation product design.

5.2.3 Characterizing Attribute Irrelevance in Product Design

For design engineers, determining how attributes within a given data set influence future consumer purchasing decisions is paramount and could mean the market success or failure of a new product. The definitions of attribute relevance presented in the previous section may not capture all of the concepts relating to product design. For example, the results in the decision tree in Figure 5.4 reveal that attributes A_1 and A_4 are not part of the decision tree and are therefore considered *irrelevant* based on the pertaining definitions of attribute relevance presented in section 5.2.2. That is, their inclusion/exclusion does not significantly influence the values of the class variable. Should attributes A_1 and A_4 therefore be omitted from future product designs and if so, what consequences would this have in the consumer market space?

To address these issues in product design, several subcategories of attribute *irrelevance* are proposed, with the goal of ensuring that vital attributes are not omitted from a product design simply based on an irrelevance characterization.

1. *Obsolete Attribute (OA):* An attribute A_i is defined as obsolete if it has been deemed *irrelevant* at iteration j (given time periods t_1,...,t_n) and its inclusion/exclusion over time does not *systematically influence* the values of a class variable. The measure of systematic influence is determined by the time series entropy trend of A_i. If A_i exhibits a monotonically increasing entropy trend (determined by the Mann-Kendall trend detection test introduced in Section 5.2.3), then this indicates that attribute A_i is consistently losing predictive power over time. If an attribute falls under this classification at the end of a given time series, it can be omitted from the next generation product designs as seen in Figure 5.5.

2. *Standard Attribute (SA):* An attribute A_i is defined as standard if it has been deemed *irrelevant* at iteration j (given time periods t_1,...,t_n) and its inclusion/exclusion over time *systematically influences* the values of a

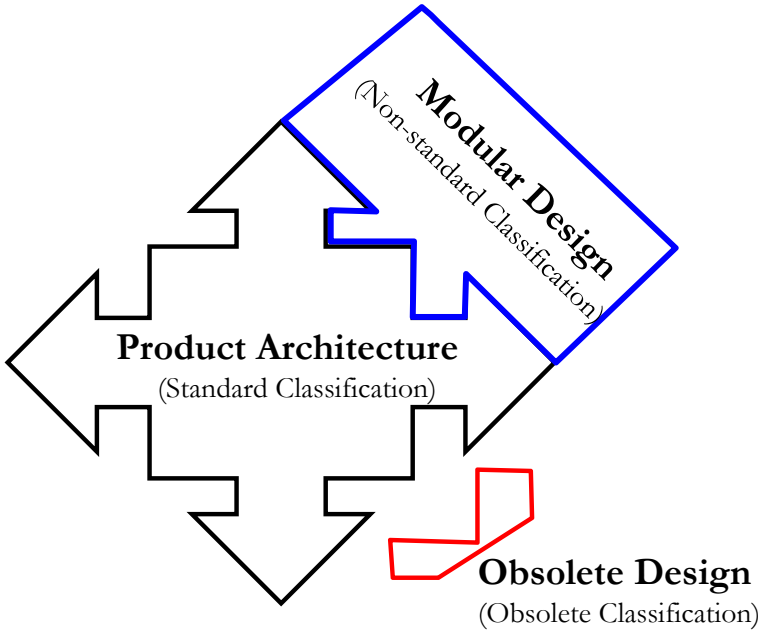


Figure 5.5: Product Design Implications of Attribute Irrelevance Classification

class variable. As with the previous definition, the measure of *systematic influence* will be quantified based on the time series entropy trend of A_i. If A_i exhibits a monotonically decreasing entropy trend (determined by the Mann-Kendall trend detection test introduced in Section 5.2.3), then this indicates that attribute A_i is consistently gaining predictive power over time (despite its initial irrelevant characterization). If an attribute falls under this classification at the end of a given time series, it should be considered vital to a product design, despite its seemingly irrelevant characterization as seen in Figure 5.5. An example of such an attribute would be an airbag in an automobile. Since almost every vehicle is now equipped with an airbag, customers may not consider this attribute while making a vehicle purchase because it is assumed to be a standard to the vehicle. If however the airbag were removed from the vehicle design, this may significantly alter a customer's purchasing decision.

3. *Nonstandard Attribute (NA):* An attribute A_i is defined as nonstandard if it has been deemed *irrelevant* at iteration j (given time periods t_1,...,t_n), and its inclusion/exclusion does not reveal a discernible relation to the class variable. This is determined by the absence of a monotonically increasing or decreasing entropy trend as determined by the Mann-Kendall trend detection test introduced in Section 5.2.3. Attributes that may exhibit this type of behavior in product design may be novel attributes that consumers may not yet fully be aware of or existing attributes that have variations within the market space. Such attributes should not be overlooked and may either turn out to be a short term consumer hype or may eventually become standard expectations. Consequently, modular components should be designed for attributes exhibiting this type of pattern (as seen in Figure 5.5) as these modules can be upgraded or eliminated all together based on future market demands.

Mann-Kendall Trend Detection

To detect trends for each Attribute A_i that has been deemed *irrelevant* at iteration j, the non parametric Mann-Kendall statistic is employed [150, 151]. The Mann Kendall trend test does not provide the algorithm with the magnitude of the trend, if one is detected. Rather, it simply quantifies the presence/absence of a trend which is all that is needed to classify each attribute within the data set. The Mann Kendall test is based on the statistic S defined as [145]:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sgn(x_j - x_i) \tag{5.5}$$

Here

- n represents the total number of time series data points.
- x_j represents the data point one time step ahead.
- x_i represents the current data point.

$$sgn = \begin{cases} 1 & \text{if } (x_j - x_i) > 0 \\ 0 & \text{if } (x_j - x_i) = 0 \\ -1 & \text{if } (x_j - x_i) < 0 \end{cases} \tag{5.6}$$

The corresponding Kendall's Tau is related to the S statistic as follows:

$$\tau = \frac{S}{\frac{1}{2}n(n-1)} \tag{5.7}$$

The null hypothesis is that there is no trend within the data. Therefore, if the resulting p-value is less than the significance level (α=0.05), the null hypothesis is rejected and a positive (positive τ) or negative (negative τ) trend is assumed.

The characterization of attribute *irrelevance* (as either obsolete, nonstandard or standard) is determined by looking beyond a single data set and generating models based on multiple time steps that quantify attribute relevance/irrelevance over time. Given a time series data set t_1 to t_n as illustrated in Figure 5.6, each data set from t_1 to t_n is analyzed and based on the *Gain Ratio* relevance definition, the test attribute A_i is characterized as either *relevant* or *irrelevant* at iteration j. If an attribute is deemed *irrelevant*, the Mann Kendall test is then employed to analyze the histories of each attribute entropy value from t_1 to t_n. An attribute value exhibiting increasing predictive power (lower entropy) over

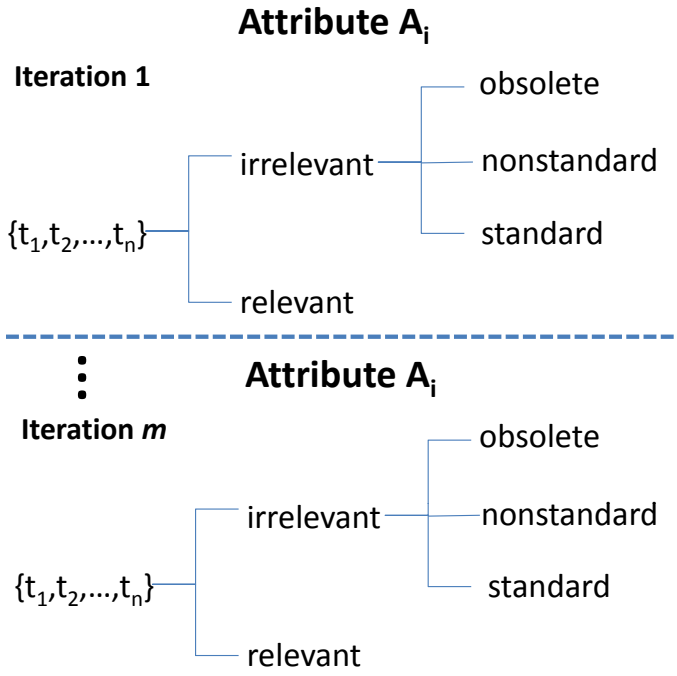


Figure 5.6: Attribute (A_i) characterization (relevant and irrelevant categorization) from iteration 1 to iteration m (each iteration contains a total of n time series data sets).

time would be deemed potentially useful in future iterations. The resulting characterization of the predictive model generated in time period t_{n+1} will therefore assign an attribute irrelevance characterization based on the trends of the historical entropy data.

Each of the attribute irrelevance definitions will be represented as a binary variable; 1 implies that an attribute is characterized as either *Obsolete* (OA), *Nonstandard* (NS) or *Standard* (SA) at a given iteration *j* and 0, otherwise. At each iteration, an attribute deemed irrelevant can only assume one of the 3 possible irrelevant characterizations. The final classification of an irrelevant attribute is assigned after the final iteration *m*. The final iteration *m* is reached after a homogeneous class distribution is attained for one of the subsets of the data (a leaf node in the decision tree structure). A variable is defined for each irrelevant characterization ($OA_{t=1,\ldots,n}$, $NS_{t=1,\ldots,n}$, $SA_{t=1,\ldots,n}$) and its value, determined by summing across all iterations (j=1,...,m) as described below:

$$OA_{t=1,\ldots,n} = \sum_{j=1}^{m} OA_j \cdot \frac{T_j}{T} \tag{5.8}$$

$$NS_{t=1,\ldots,n} = \sum_{j=1}^{m} NS_j \cdot \frac{T_j}{T} \tag{5.9}$$

$$SA_{t=1,\ldots,n} \quad = \quad \sum_{j=1}^{m} SA_j \cdot \frac{T_j}{T} \tag{5.10}$$

Here,

- T_j: represents the number of data instances used to calculate the *Gain Ratio* statistics at iteration j.
- T: represents the total number of data instances in the entire data set.

At iteration j, each attribute characterization is weighted based on the the proportion (T_j/T) of instances. Therefore the initial characterization at iteration 1 (containing the entire data set) carries the most weight due to the presence of all instances of the data. The classification of an attribute at time step t_{n+1} is determined by selecting the irrelevant characterization with the highest variable value (($OA_{t=1,\ldots,n}$, $NS_{t=1,\ldots,n}$, $SA_{t=1,\ldots,n}$)). Given time steps t_1,...t_n, the pseudo code for the irrelevant attribute characterization for Attribute A_i is as follows:

1. `Start: Iteration j=1`
2. `If predicted` *`Gain Ratio`* `of Attribute` A_i `is not the highest, Attribute` A_i `is considered irrelevant`
3. `Employ Mann Kendall (MK) trend test for Attribute` A_i
4. `If MK` τ `is negative (with p-value < alpha), irrelevant classification=Standard`
5. `Else If MK` τ `is positive (with p-value < alpha), irrelevant classification=Obsolete`
6. `Else If MK` τ `is positive/negative (with p-value > alpha), irrelevant classification=Nonstandard`
7. `While data set/subset does not contain a homogeneous class`
8. `Split the data set into subsets based on the number of mutually exclusive values of the attribute with the highest Gain Ratio from Step 2`
9. `j=j+1 and revert to Step 2 for each data subset`
10. `End Tree, Classify Irrelevant Attribute` A_i `based on highest variable value ((`$OA_{t=1,\ldots,n}$`,` $NS_{t=1,\ldots,n}$`,` $SA_{t=1,\ldots,n}$`))`

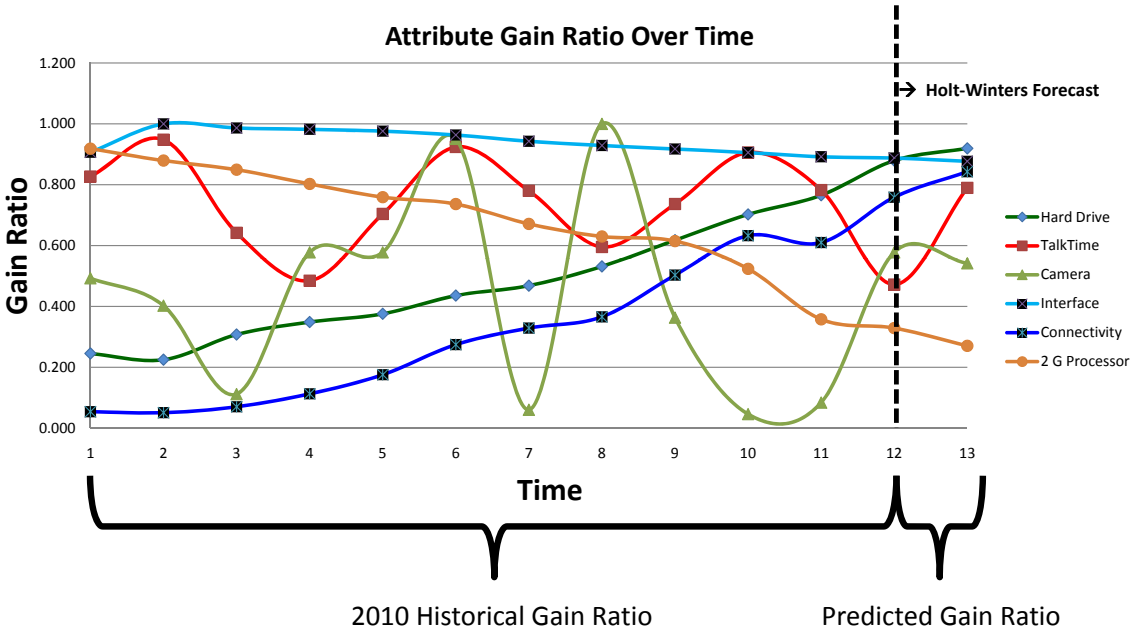


Figure 5.7: Time Series *Gain Ratio* at iteration 1 (Period 1-12 with Period 13 predicted by employing the Holt Winters predictive model)

Product Concept Demand Modeling

Once the time series decision tree model has been generated and irrelevant attributes characterized, a fundamental question that still remains is how to estimate the demand for the resulting product concepts (unique attribute combinations). For example, the resulting product concept {**Hard Drive**=16GB, **Interface**=Slider, **Price**=$179} in the left branch of Figure 5.9, enterprise decision makers would want to know the overall market demand for this particular product so that potential product launch decisions can be made. With a traditional decision tree model (using a static data set for model generation), the demand for this particular product concept will be a subset of the original training data set used to generate the model (T_m/T, where T_m denotes the number of supporting data instances after m iterations/data partitions) [29]. This is analogous to a product's *choice share* (Discrete Choice Analysis case) which has been used extensively by researchers in the design community to estimate product demand [141, 152, 153]. Since the proposed Trend Mining algorithm is making predictions about future product designs, the demand for a resulting product concept is estimated based on the time series trend of the supporting instances T_m using the Holt-Winters forecasting approach presented in section 5.2.1. This will enable design engineers to anticipate future product demand for the predicted trend mining model.

5.3 Product Design Example

5.3.1 Cell Phone Design Study

To validate the proposed trend mining methodology, several well known data sets (from the UC Irvine machine learning repository) are tested to compare the results of the proposed Preference Trend Mining algorithm with traditional demand modeling techniques. For conciseness, a detailed explanation of the cell phone case study is presented while only providing the results for the remaining data sets used in the evaluation. The original cell phone case study was based on a University of Illinois online survey of cell phone attribute preferences originally created using the UIUC webtools interface [55, 29]. To accommodate the time series nature of the proposed methodology, the product design scenario is presented as follows:

Enterprise decision makers within a cell phone company are looking to launch their next generation cell phone early in the first quarter of 2010. To guide their product design decisions, 12 data sets (representing monthly customer preference data for fiscal year 2009) are available through online customer feedback. Based on the time series data, design engineers want to integrate customer preferences directly into the next generation product design. The goal of the new cell phone project is for the functionality of the next generation cell phone design to anticipate the preferences of the customers at the time of product launch; preferences that are constantly evolving within the market space.

For each monthly data set, there are 6 product attributes and 1 dependent variable. There are a total of 12,000 instances (customer response) for the entire 12 month time period, partitioned into 1000 instances of customer feedback per month. The attributes, along with their corresponding values are as follows:

- **Hard Drive:** {8 GB, 16 GB, 32 GB}
- **Talk Time:** {3 Hours, 5 Hours, 7 Hours}
- **Camera:** {2.0 MP, 3.1 MP, 5.0 MP}
- **Interface:** {Flip Phone, Slider Phone, Touch Screen Phone}
- **Connectivity:** {Bluetooth, Wifi}
- **2G Processor:** {Limited, Capable}

The class variable is the price category of the given cell phone design within the time series data: **Price:** {$99, $149, $179, $199, $249}.

The structure of the data is similar to that presented in Figure 5.2 with the attribute names indicated by the first row of each column (except for the last column which represents the class variable, price). In the time series data, the distribution of the attributes as well as the class values associated with each attribute value changes over time.

Up until now, demand modeling in product design had focused on utilizing the most recent data set to generate predictive models about future customer behavior. The research findings presented in section 5.4 reveal that such techniques may not fully capture emerging consumer preference trends and may ultimately mislead future product design decisions.

5.4 Results and Discussion

The results of the cell phone case study introduced in section 5.3 provide valuable insight into the challenges of designing products for volatile consumer markets. To begin, the time series *Gain Ratio* statistics for each attribute (at iteration 1) shown in Figure 5.7 is presented. In the proposed trend mining methodology the algorithm takes into consideration all possible scenarios for the attribute *Gain Ratio* statistics over time. The algorithm captures attributes that display a monotonically increasing or decreasing trend, a seasonal trend or no trend at all which is model using the Holt Winters technique presented in section 5.2.1. Based on the level of seasonality or trend within the data, the one time step ahead predictions (period 13) are modeled. At period 12 in Figure 5.7, it can be observed that the *Interface* attribute has a higher *Gain Ratio* than the *Hard Drive*. However, based on the emerging trends of these two attributes, it can be observed that the *Hard Drive* attribute will have a higher *Gain Ratio* in future time periods, which the Holt Winters model predicts in time period 13.

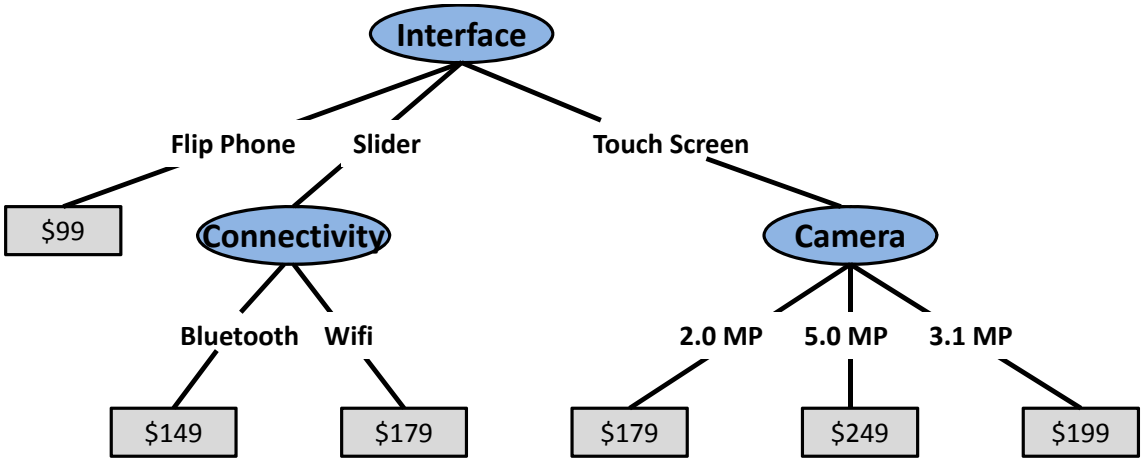
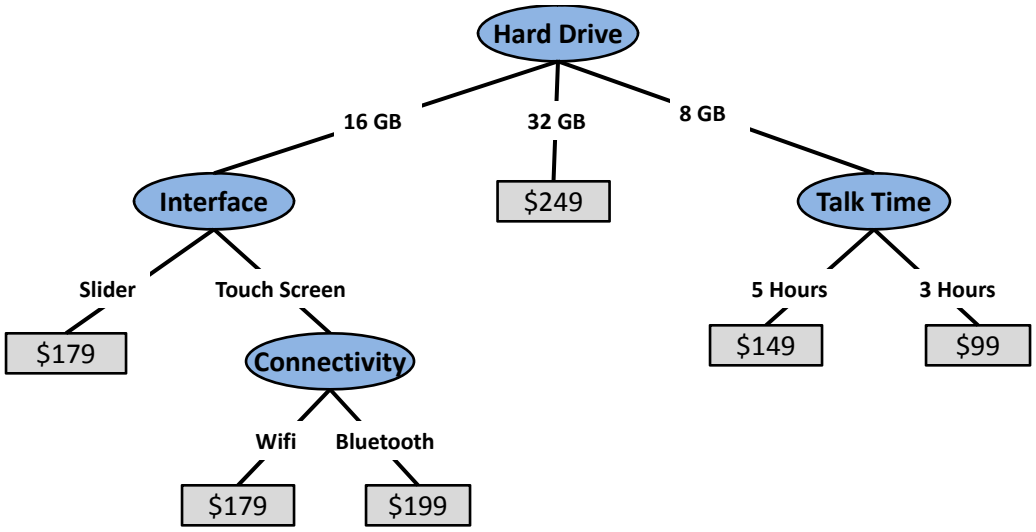


Figure 5.8: Decision Tree Model using Period 12, 2009 data set only for model generation(results attained using Weka 3.6.1 [2])

New design insights obtained by preference trend mining In order to understand the product design implications of these findings, Figure 5.8 presents the predictive model results that are generated using the most recent data set (period 12). In Figure 5.8, the only *relevant* attributes to the price variable are: *Interface*, *Connectivity* and *Camera*,



Tree Path (right to left)	Relevant Attributes	Price	Demand	Irrelevance Characterization		
				Standard	Nonstandard	Obsolete
1	Hard Drive, Talk Time	$99	180	Camera	2G Processor, Connectivity	Interface
2	Hard Drive, Talk Time	$149	362		2G Processor, Connectivity	Interface, Camera
3	Hard Drive	$249	73	Talk Time, Camera	2G Processor, Connectivity	Interface
4	Hard Drive, Interface Connectivity	$199	412	Talk Time, Camera	2G Processor	
5	Hard Drive, Interface Connectivity	$179	99	Talk Time, Camera	2G Processor	
6	Hard Drive, Interface	$179	208	Talk Time, Camera	2G Processor, Connectivity	

Figure 5.9: Trend Mining Model using Periods 1-12, 2009 data for model generation (results attained using ESOL developed Java Based PTM code compatible with Weka [2])

with the associated decision rules acquired by traversing down the appropriate paths of the decision tree. In contrast, when the proposed time series Preference Trend Mining algorithm is employed using the data from periods 1-12, there are noticeable differences in the resulting attributes that are considered *relevant* (Figure 5.9). From the resulting decision trees in Figures 5.8 and 5.9, it can be observed that the common attributes between the two models are the *Interface* and *Connectivity* attributes. However, even with the *Interface* attribute being common between the two models, it can be observed that the *Flip Phone* interface design found in Figure 5.8 is not included in Figure 5.9, providing engineers with the knowledge that this particular attribute value is not desired in future time periods. Given the differences between these two decision tree structures, entirely different product design decisions may result to address the needs of the market.

Furthermore, for those attributes that are considered *irrelevant* to the classification of price (and are therefore omitted from the decision tree model in Figures 5.8 and 5.9), design engineers have no direct way of deciding whether these attributes should be omitted from all future cell phone designs. As a reminder, an *irrelevant* attribute simply

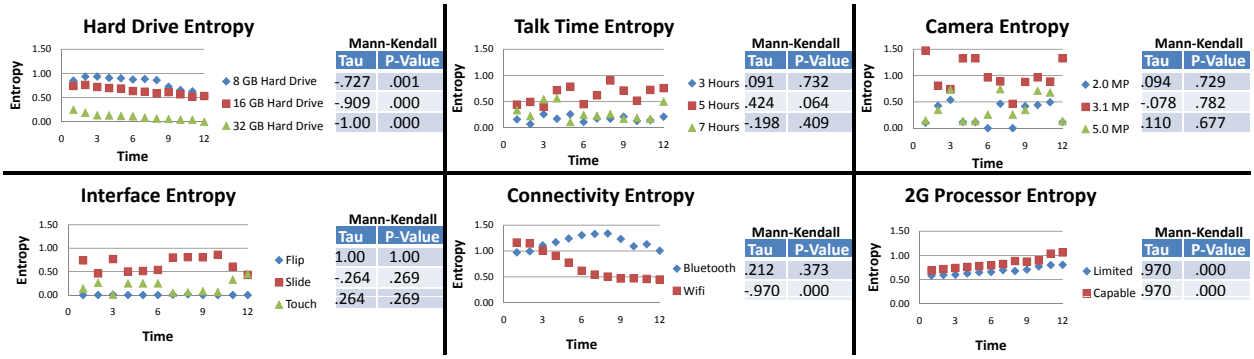


Figure 5.10: Time Series Attribute Entropy values for irrelevance characterization

Table 5.2: Comparison of predictive accuracies between the PTM and DT models using Time Series Data

Predictive Model	Model Validation Characteristics						
	Data set	# Attributes	#Instances/Period	# Periods to Train	# Periods to Test	Higher % Accuracy	P-Value
PTM DT	Car Evaluation	7	1728	24	12	x	0.00507
PTM DT	Cylinder Bands	10	540	36	24	x	0.00007
PTM DT	Automobile Brand	9	205	24	12	x	0.00008

means that at iteration j, an attribute does not have the highest *Gain Ratio*, not necessarily that it does not have any predictive power whatsoever, as illustrated in Figure 5.7. At iteration 1, since the PTM algorithm predicts that the *Hard Drive* attribute will have the highest *Gain Ratio* at time period 13 (see Figure 5.7), the remaining attributes are characterized as either *obsolete*, *nonstandard* or *standard*. The entropy histories along with the results from the Mann Kendall trend test in Figure 5.10 indicate that the 2G Processor is characterized as *obsolete* (positive τ values and p value within tolerance limit) while the remaining attributes are characterized as *Nonstandard* (due to p values exceeding the tolerance limit). After subsequent iterations of the PTM algorithm, the attributes that do not show up in the tree are therefore classified as shown in Figure 5.9, with the accompanying demand (# supporting predicted instances) accompanying each branch of the tree.

5.4.1 Model Validation

In addition to the structural differences of the resulting decision tree models, there are also noticeable differences in the predictive accuracies. Figure 5.11 presents the predictive accuracy results between the proposed Preference Trend Mining (PTM) model and the traditional Decision Tree (DT) classification model. The predictive accuracies are calculated using 12 monthly data sets from 2010. For each instance in a given monthly data set, the attribute combinations resulting in a class value are tested against the decision tree predictions by traversing down the path of the decision trees in Figures 5.8 and 5.9. If the class value predicted by the decision tree model matches the actual

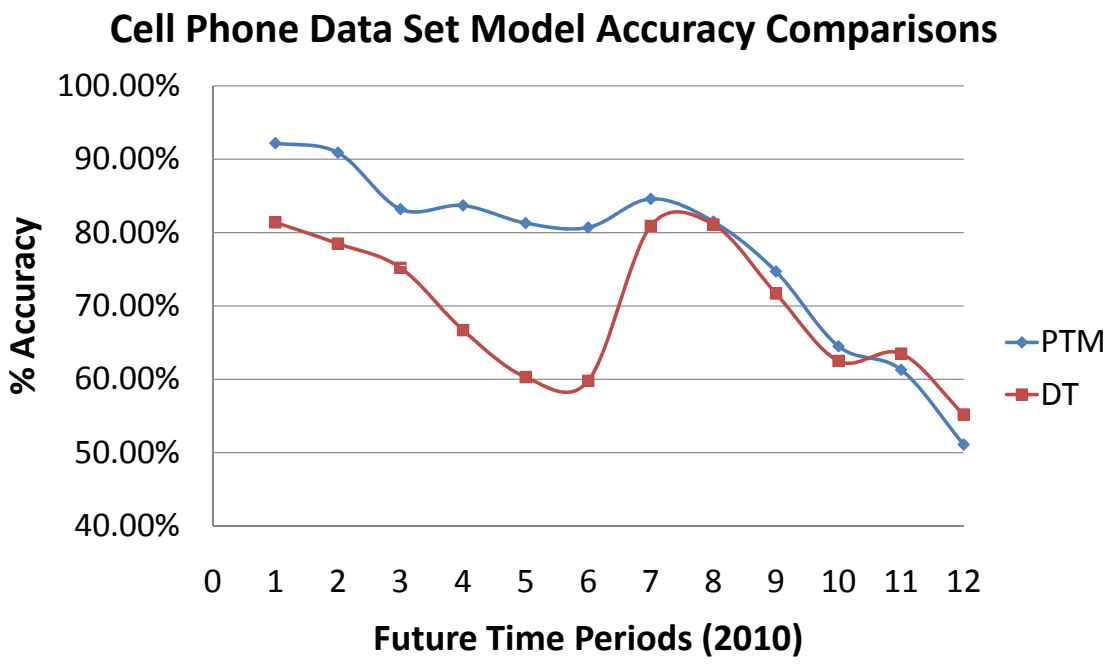


Figure 5.11: Comparison of predictive accuracies between the PTM and DT models (using 12 unseen time stamped data from 2010 [2])

class value in the monthly data set, a value is incremented in the *correct predictions* category, otherwise a value is incremented in the *incorrect predictions* category. The summary predictive accuracies in Figure 5.11 reveal that the PTM model attains a higher predictive accuracy for many of the time periods, compared to the DT model.

To obtain a statistically valid conclusion on the predictive accuracies of the two models, the Wilcoxon signed rank test is employed which has been proposed in the data mining/machine learning literature as a suitable approach for comparing two models against multiple data sets [154, 34]. The null hypothesis of the test is that the median difference between the two model accuracies is zero. The alternate hypothesis is that the accuracy of the DT model is less than that of the PTM model. Using a significance level of α=0.05, the null hypothesis (data in Figure 5.11) is rejected with a p value of 0.0224, providing statistical evidence that the accuracy of the PTM algorithm exceeds that of the DT for the Cell Phone data set. It can be seen that the predictive accuracy of both models diminishes over time with values slightly above 50 % in period 12. The PTM accuracy may be enhanced in future time periods by changing the k value of the k-ahead time predictions from 1 (in the cell phone model) to 12.

Additional data sets from the UC Irvine machine learning repository were generated and tested against the two models for model accuracy with varying data set conditions (attribute space, number of instances, number of time periods, etc.) with the results presented in Table 5.2. The results from Table 5.2 emphasize the robustness of the proposed PTM algorithm in handling different types of time series data while still maintaining greater predictive

accuracies, compared to the traditional decision tree model. Due to the variation in data set structure, size, etc., it is rare for an algorithm to outperform on every metric of performance [155]. Therefore, the proposed PTM model is well suited for data sets that exhibit monotonically increasing/decreasing or seasonal trends similar to the test data sets presented. In scenarios where no discernable trends exist in the data set, the PTM algorithm was found to perform comparable to traditional demand modeling techniques which should not be surprising, given the underlying formulation of the proposed PTM algorithm.

5.5 Conclusion and Path Forward

The major contribution of this research is to propose a machine learning model that captures emerging customer preference trends within the market space. Using time series customer preference data, a time series exponential smoothing technique is employed that is then used to forecast future attribute trend patterns and generate a demand model that reflects emerging product preferences over time. The Mann Kendall statistical trend detection technique is then used to test for attribute trends over time. An attribute irrelevance characterization technique is also introduced to serve as a guide for design engineers trying to determine how the classified attributes are deemed irrelevant by the predictive model. The insights gained from the preference trend mining model will enable engineers to anticipate future product designs by more adequately satisfying customer needs. Future work in customer preference trend mining will include expanding the current approach to handle the continuous attribute and class domain.

Chapter 6

Capturing Product Preference Trends Using Publicly Available Customer Review Data

6.1 Introduction

A major challenge that has plagued the product design community has been the lack of large scale, realistic customer data to validate proposed product design methodologies or solve complex design problems. More than often researchers and design engineers resort to conducting customer surveys or focus group interviews that can be costly and time consuming. Recent research studies have shown that a major online retail company can have as many as 10 million active product reviews while it has been reported that more than 50% of online customers indicated that customer reviews played an important role in influencing their purchasing decisions. Based on these findings, the authors of this work have developed a publicly available online interface (www.trendminingdesign.com) built upon an automated computer exploration algorithm that captures and stores time series product preference data. The goal of this research is to enhance the design creativity process by providing design engineers with direct access to large scale, time series customer preference data that can be used to guide the product concept generation process.

The rapid expansion of internet usage, both domestically and globally, is helping fuel an increasing flow of information across many barriers. Technological successes such as Wikipedia, Facebook®, Twitter®, etc., are giving users a sense of empowerment in their ability to create and shape knowledge and information flow. These user-propelled networks have been referred to as *digitized word of mouth* networks that harness the true power of human communication [156].

Online customer reviews are becoming a viable source of large scale product review data. A recent research study found that Amazon.com had over 10 million active customer reviews on all product categories [157]. A study by Forrester Research reported that 50% of customers who visited retailer sites with customer review feedback indicated that customer reviews were important or extremely important in their purchasing decisions [157].

Many research domains such as Marketing and Advertising, Medical Research, Quality Assurance, Computer Science [158, 159],etc., continue to investigate the potential of web based networks as a viable approach to data collection. However, research into the acquisition and integration of online information in the product design domain

has been limited.

A few of the noticeable benefits of employing such approaches in customer data acquisition include the speed at which large sets of customer review data can be accessed and stored for next generation product design purposes. Another major benefit is that a significant portion of this data is based on customer *revealed preferences*; that is, the feedback given after a customer has purchased and interacted with a product for a considerable amount of time. This differs from *stated preference* data that typically involves a hypothetical purchasing scenario in the form of a survey [29].

The methodology presented in this work (1) Searches through user specified customer review web sites for a given product; (2) For each unique user review, a text mining and retrieval algorithm is employed to isolate and store information specific to a given product review; (3) The stored text data for the entire review population is time stamped and mined for frequent feature patterns; (4) A time series predictive model is generated based on a specific time horizon; and (5) Design engineers can utilize the generated model to understand evolving consumer preference trends for next generation product design ideas.

6.1.1 Traditional Customer Preference Acquisition Techniques

There are several well established methodologies in the product design community that have been employed in the product design and development process. The Quality Function Deployment (QFD) is a product design methodology that attempts to translate customer requirements (CR) (also known as the Voice of the Customer (VOC)) into functional engineering targets in an effort to generate new design ideas and enhance quality[11] .

Another popular approach to customer preference quantification is the Discrete Choice Analysis (DCA) technique, which includes the Probit Model and Logit Models (multinomial, mixed, nested, etc) to name but a few. By quantifying product attribute levels, design engineers can estimate the demand of next generation products and also investigate creative approaches to help address customer preferences [160].

A major challenge resulting in the aforementioned customer preference modeling techniques is that the quality of the customer feedback is highly dependent on the framing of the survey questionnaires presented to current/potential customers. Recent research investigating the validity of customer surveys have revealed that a phenomenon called *self-generated validity effects* may adversely affect the quality of the customer response data [161]. Consequently, the creativity of the engineering design solution may be adversely affected if customer needs are not fully understood. For example, design engineers may be focused on developing creative approaches to enhancing the speed of a product while the emerging needs of the customer are more aligned with environmental safety. In addition to these complexities, the size of the survey data set is often quite limited due to the time and financial costs of generating surveys.

To alleviate some of these challenges, a customer preference acquisition model is proposed that is based on large scale online customer review data. The *unstructured* nature of online customer review data relieves respondents from the traditional predefined structure of a survey type approach and enables respondents to provide an unbounded assessment of their product preferences. The next section provides some background literature in the text mining research domain.

6.1.2 Online Customer Preference Acquisition Techniques

Considerable research has been done in both document summarization and text classification relating to individual words or group of words and their descriptive relations [162]. While many of the early text extraction and mining algorithms focused on document summarization, there have been several works more closely related to feature extraction as it relates to customer review data [163, 164]. Jinal and Lui investigate comparative sentence mining and employ sequential rules to compare customer sentiments towards similar products[165]. Dave *et al* propose a feature selection classification algorithm that analyzes customer review data and automatically partitions words into positive and negative domains with relatively high accuracy [162]. A more recent contribution by Hu and Liu builds upon the work by Dave *et al* by proposing a semantic classifier of product review sentences that does not need a set of training texts to build the classifier [166]. It has been reported in the literature that attempting to mine customer review data in order to separate positively and negatively associated words can be very challenging [167]. Although many methodologies have been proposed to try and address this issue in text mining, the proposed methodology overcomes these inherent challenges of text approximation by focusing on customer review sites that have partitioned the reviews into predefined categories of *pros* and *cons*.

Another related field of text mining is search query analysis. In a recent research finding, online search queries were used to predict seasonal flu patterns within the same geographical region [158]. That is, there was a direct correlation between the temporal frequency of certain key query words that describe a flu (for example, fever, aches, sneezing, etc.) and the number of hospital visits for flu like symptoms [158].

The methodology presented in this work differs from the aforementioned product review based algorithms by assessing individual customer reviews (in its entirety) and employing a text classification algorithm to determine the most relevant product features being expressed by customers. Since each user review is time-stamped, the temporal nature of certain positive and negative product features can be quantified, hereby enabling engineers to model product feature preference trends.

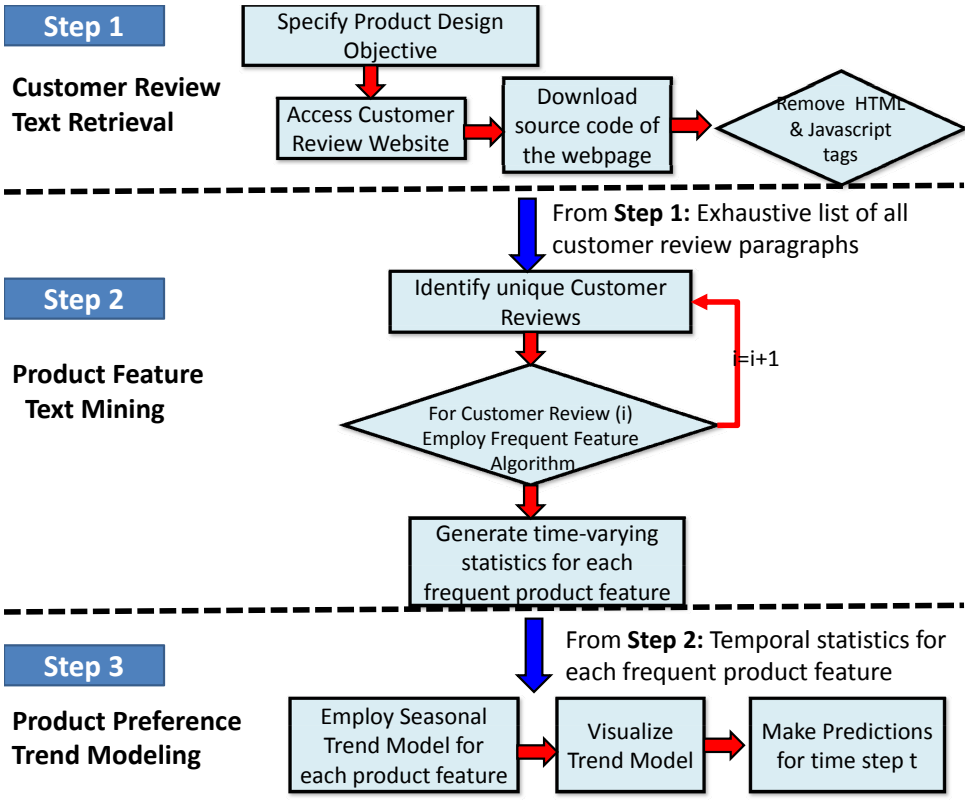


Figure 6.1: Overall flow of from Customer Review Text Retrieval to Product Preference Trend Modeling

6.2 Methodology

The proposed methodology enables design engineers to model product trends and identify emerging features that may be popular in future product design models or identify obsolete features that should be excluded from future product designs. By integrating free, publicly available customer review data, the proposed design methodology can have wide applicability to many areas of product design. The flow diagram in Figure 6.1 presents the overall proposed framework from customer review text retrieval to product preference trend modeling. The design creativity implications of each step of the flow diagram in Figure 6.1 will be presented in the following section.

6.2.1 Step 1: Customer Review Text Retrieval

The process of acquiring text based data begins by specifying the source of the customer review data. Figure 6.2 presents a snapshot of a typical cell phone customer review with the accompanying *html* source code. To overcome some of the text analysis challenges discussed in section 6.1.2, data is acquired through customer review websites that have a predefined partition of the positive (pros) and negative (cons) customer reviews (ex:www.cnet.com). The pre-partitioned format of the customer feedback platform greatly reduces the complexities that would have resulted

from trying to mine the raw data for negative and positive customer opinions.

Figure 6.2: Text Summarized Customer Review Data (Partitioned into Pros and Cons)

The authors of this work have developed a product design research website www.trendminingdesign.com that acquires all of the raw customer review data (in html source code format) from multiple online sources.

Implications for product design creativity

The unconstrained nature of online customer review data is a departure from traditional survey type approaches and has the potential to enhance the overall product design process by enabling design engineers to understand the entire spectrum of customer needs. The formulation of a survey questionnaire, ironically assumes that the customer cares about the the specific items presented within the survey. In addition to this bias, it has been reported in the literature that an individual's reported purchase intentions is biased towards a social norm whenever they are asked to make predictions about their future behavior (ex: An individual may express preference towards a green product due to social norms, despite whether or not this product will satisfy their needs) [161]. By extracting customer preferences through an unguided web-based format, design engineers can acquire large scale unpredictable, honest customer feedback in a timely and efficient manner. There are however some fundamental challenges of acquiring unstructured customer review which are addressed in Step 2 of the product design process.

6.2.2 Step 2: Product Feature Text Mining

Customer review data in html format is stored on an SQL database on www.trendminingdesign.com. The next step is to determine the product features being expressed by customers. This is a non-trivial problem as customer reviews are presented in an unstructured, unpredictable style. The methodology beings by employing a PHP: Hypertext Preprocessor (PHP) based version of the Brill Tagger for each customer review sentence [168]. This enables each customer

review sentence to be decomposed into their respective Part of Speech (POS), otherwise known as Part of Speech Tagging. For example, the customer feedback "Easy user interface" presented in Figure 6.2 would be transformed into:

- [easy-**JJ** user-**NN** interface-**NN**]

where **JJ** is the Tag code for an *adjective* and **NN** is the Tag code for a *noun*, with similar syntax used for other parts of speech. The results of the Tagger algorithm will therefore identify the product feature space within customer reviews that are represented by nouns. For each time step, a frequent word algorithm searches through customer reviews to identify the most frequently expressed customer product preferences. The sensitivity of the frequent words search algorithm is dependent on the user specified minimum threshold (MinSup) for what is characterized as *frequent*.

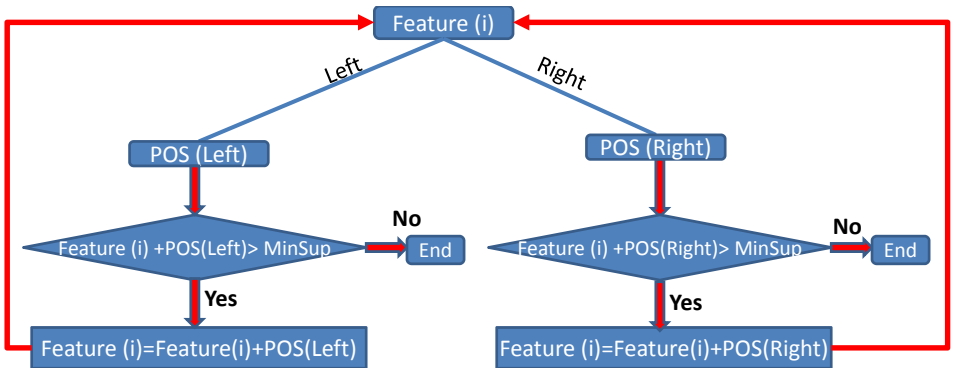


Figure 6.3: Algorithm flow of product feature search aggregation

One major challenge in refining the accuracy of the frequent product feature algorithm is to efficiently analyze and understand customer textual input. For example, the Tagger results above reveal that both the words *user* and *interface* are nouns, representing candidate product features. However from observation, it is evident that the customer is referring to a single product attribute *user interface*, rather than two separate attributes *user* and *interface*. An Apriori-like algorithm is employed that satisfies the anti-monotone Apriori property: *if any length k pattern is not frequent in the database, its length (k+1) super-pattern can never be frequent*[169, 127]. Figure 6.3 presents a visual flow of the frequent product feature extraction algorithm that enables words like *user* and *interface* to be grouped as one product feature *user interface*.

Once the frequent features have been identified for each time step, design engineers may want to know the *subjective* and *objective* terms that customers use to describe them. An example of a subjective descriptive term would be the word *easy*, used to describe the product feature *user interface*. An example of an objective description would be *16 Gigabytes* used to describe the product feature *hard drive*. A Bayesian Classification learner is employed to

determine the associated terms (objective or subjective) relating to the most frequent features. This is mathematically represented as:

$$p(W_j|F_i) = \frac{p(F_i|W_j) \cdot p(W_j)}{p(F_i)} \tag{6.1}$$

where,

- F_i: represents a particular product feature.
- W_j: represents the particular word (adjective, adverb, etc.) describing feature F_i.

This allows engineers to classify a particular descriptive term (W_i) based on the probability estimates from equation 6.1 by assigning the objective/subjective term to the feature with the highest probability.

The association of customer sentiment with a given product feature allows engineers to transform the once unstructured customer review data into a high dimensional structured representation that can be used for traditional statistical analysis and data mining in the subsequent product design process [55, 29].

Customers	Feature 1	...	Feature N
Customer 1	W_1	-	-
.	-	-	W_7
.	-	W_9	W_{11}
.	W_3	-	-
Customer M	-	-	W_8

Figure 6.4: Transformation of unstructured customer review text data to structured customer feature preference data

This structured format can be downloaded directly from the research website www.trendminingdesign.com where engineers can search through and access their product of interest. It is important to note in Figure 6.4 that not all feature words (W_i) are present for each customer, which is a more realistic representation of market conditions as different customers would express varying preferences of product features.

Implications for product design creativity

The ability to identify the positive and negative product features (over time) most frequently expressed by customers will serve as a valuable feedback tool in the product design process. Design engineers will be able to enhance the technological features that customers respond positively towards and either substitute or eliminate product features

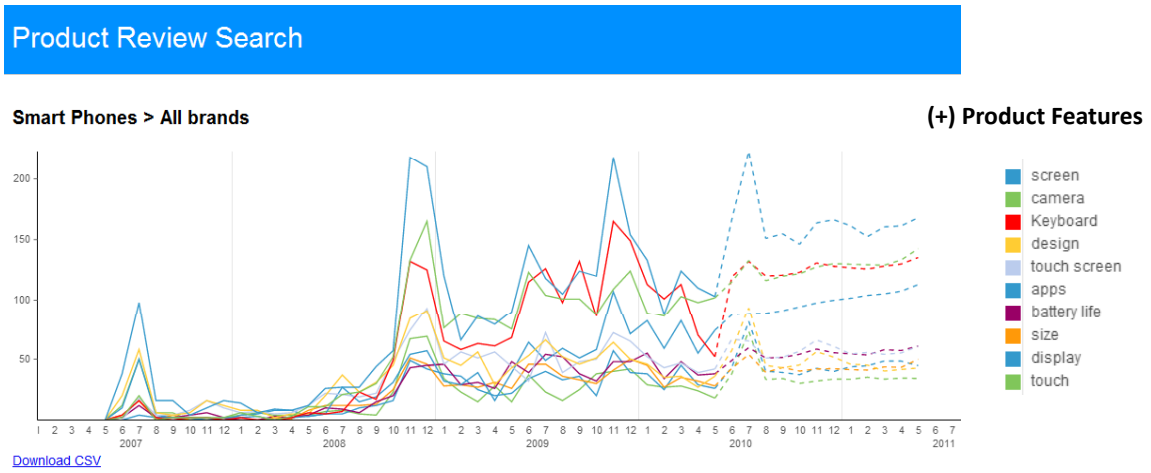


Figure 6.5: Plot of product preference trends with Holt-Winters forecasting

receiving significant negative reviews. Equally as important as the feedback is the magnitude and speed at which this feedback can be acquired using the proposed online customer review process. Currently, there exists more than 17,000 customer reviews on the website www.trendminingdesign.com that spans more than 400 product models. Therefore enterprise decision makers will not only be able to acquire reviews about their own products, but also benchmark their electronic feedback to that of competitors also being discussed within the customer review platform. All of this insight can be acquired in a few seconds which it takes to generate a set of time series frequent product feature results.

6.2.3 Step 3: Product Preference Trend Modeling

The Holt Winters exponential smoothing technique is employed to model the time series data due to its relatively high predictive accuracy compared to other models such as the STL, Box-Jenkins, AutoRegressive Integrated Moving Average (ARIMA), to name but a few [143, 147]. The model uses a weighted averaging technique that takes into account the local level, the trend, and the seasonal components of the time series. The (k) step-ahead forecasting model can therefore be represented as:

$$\widehat{y}_t(k) = L_t + kT_t + I_{t-s+k} \tag{6.2}$$

where

Level L_t (the level component):

$$L_t = \alpha(y_t - I_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{6.3}$$

Trend T_t (the slope component):

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1} \tag{6.4}$$

Season I_t (the seasonal component):

$$I_t = \delta(y_t - L_t) + (1-\delta)I_{t-s} \tag{6.5}$$

The smoothing parameters α, γ, δ are in the range $\{0,1\}$ and are typically chosen to be anywhere from 0.02 to 0.2 as a default, although they can also be estimated by minimizing the sum of squared errors for one time step ahead [143, 147]. The starting values L_0, T_0 and I_0 also have to be initiated with conventional estimates [143].

Implications for product design creativity

By generating predictive models to forecast product feature trends, design engineers can observe which product design features are becoming obsolete or popular over time and incorporate these findings in future product design decisions. A visual representation of the trend mining model is presented in Figure 6.5 with the solid lines representing the historical data acquired from customer reviews and the dashed lines representing the Holt-Winters forecasts beyond the actual data. The graphical predictive model enables engineers to analyze the customer preference trends of the entire product industry (ex: entire smart phone market in Figure 6.5) or isolate a particular product type for analysis. Taking a closer look at Figure 6.5, it can be observed that in June, 2007, the product feature *Keyboard* has a relatively low preference. Moving thought time however, it can be observed that the *Keyboard* feature becomes the 3rd most popular feature expressed by customers.

Design Engineers also have the ability to benchmark their product to competitors by comparing the time series product features to those of competitors. Figure 6.6 presents a scenario where the *negative* product preferences are compared between two product brands, Apple and HTC. Figure 6.6 reveals that the negative product feature opinions of the two brands begin to differ at the 3rd highest negative product feature camera (for Apple) and screen (for HTC). Design engineers can incorporate this negative customer feedback into the design of next generation products by developing creative solutions to help address product deficiencies and minimize the threat of competitors. One of the major benefits of modeling customer preference trends using online, publicly available data is that models can be

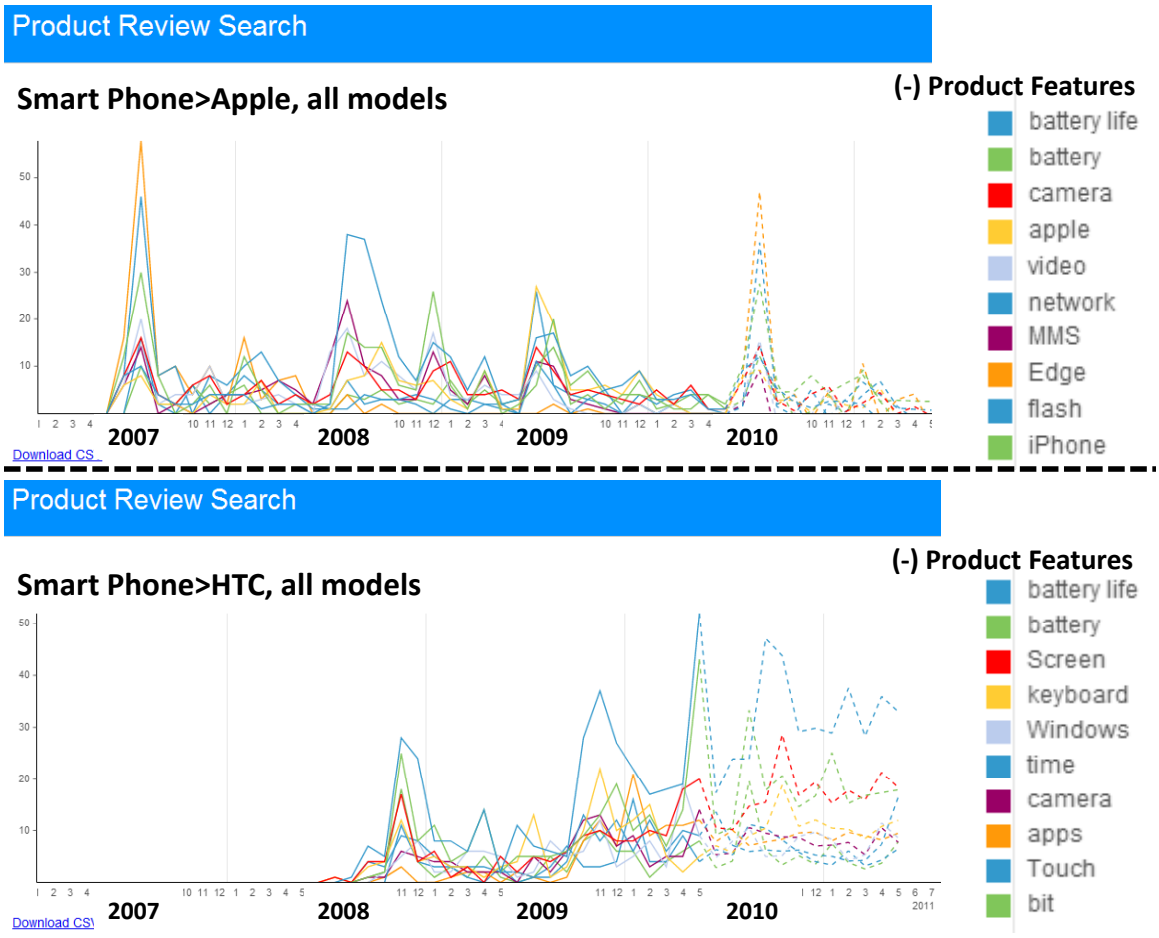


Figure 6.6: Trend Comparison of negative product features between Apple and HTC

frequently updated as new customer reviews are accessed and mined. This data is readily available for download in a structured .CSV format as seen from the option in the bottom left of Figure 6.5.

6.3 Conclusion

In this dissertation, the product design challenge of large scale customer data acquisition has been addressed by proposing an online, publicly available customer product review methodology presented on the web site www.trendminingdesign.com. A product design methodology is proposed that transforms unstructured customer preference data into a time series representation of product feature preferences. Although the primary focus is on consumer electronics, the proposed methodology can be extended to other engineering fields such as automotive design, aviation logistics, etc. The aim to expand on the proposed online trend mining research by developing demand models for next generation products based on the time series customer review data.

Chapter 7

Conclusion and Future Work

The advent of low cost data storage resources and multiprocessor computing machines has made the acquisition and storage of data more accessible. However with the increase in data storage comes the non trivial challenge of extracting useful, relevant knowledge. Machine learning techniques have been proposed to help address some of these emerging data intensive problems.

The foundation of this research is to develop a Multidisciplinary Design Optimization (MDO) approach to product design by creating a synergistic methodology that incorporates the objectives of each discipline (engineering design, manufacturing, distribution, etc.) into the realization of an optimal product portfolio. With the incorporation of predictive data mining/machine learning techniques, this research merges customer preferences directly with engineering design to achieve the most sustainable product portfolio. This process is an iterative approach that employs the decomposition and integration techniques of multilevel optimization.

The focus of this research has been to explore how machine learning techniques can be used to solve complex systems design problems in a timely and efficient manner with the main focus being product design problems. Traditional data acquisition techniques that have been employed in the product design community have relied primarily on customer survey data or focus group feedback as a means of integrating the customer preference information into the product design process. Direct customer interaction can be costly and time consuming and may therefore limit the overall size and distribution of the customer preference data. Furthermore, since the survey data acquired typically represents stated customer preferences (customer responses for hypothetical product designs, rather than actual product purchasing decisions made), design engineers may not know the true customer preferences for a new product design.

The Preference Trend Mining (PTM) algorithm that is proposed in this work aims to address some fundamental challenges of current engineering product modeling techniques by capturing changes in product preferences over time. This enables design engineers to anticipate next generation product features before they become mainstream/unimportant.

Although this research has focused primarily on machine learning in the context of product design, research extensions can be made to other complex systems design problems involving large scale data generation.

7.1 Future Work

7.1.1 Capturing Emergent Behavior in Systems of Systems Modeling

A potential future expansion could be the application of the aforementioned machine learning approaches to Systems of Systems research through collaborations with Sandia National Laboratories involving their Future Combat Systems (FCS) analysis models. Systems of Systems research involves the integration of multiple large scale, complex systems for enterprise level decision making. While there are several definitions of Systems of Systems, many share certain similarities such as Operational Independence, Managerial Independence, Evolutionary Development, Emergent Behavior, and Geographical Distribution of each independent system. Simulations involving Systems of Systems models often generate large scale data that can be difficult to analyze in a timely and efficient manner. The focus of this research will be to propose efficient methodologies that help capture emergent behavior of systems of systems models and aid enterprise decision makers in the initial design and decision making strategies. Mechanical engineering systems that exhibit systems of systems characteristics will serve as the basis for model validation.

7.1.2 Sustainable Product Portfolio Design

Future research aspirations aim to propose novel machine learning algorithms for sustainable systems design. As engineers and researchers, the definition of *product optimality* should be expanded beyond the traditional objectives of cost and performance to include measures that evaluate designs based on the long term environmental impact, energy requirements or the ability to be reused or recycled. There are many facets of sustainable engineering that not only include the discovery of breakthrough, low cost energy sources, but also the efficient design and disposal of products. This research project will comprise of both simulation based design and prototype testing of revolutionary clean technologies to help address the aforementioned global air quality challenges. Instead of designing separate systems that can enhance the sustainability of an engineering artifact, this project will investigate the feasibility of designing large scale manufacturing systems with sustainable modules/components such as air purification systems that may increase the operating life of machinery, while reducing carbon footprint.

7.1.3 Cyberinfrastructure in Multidisciplinary Design Optimization

Cyberinfrastructure plays a significant role in addressing the complex challenges often faced in the engineering design community. This research will investigate the benefits of Cyberinfrastructure as it relates to Multidisciplinary Design Optimization. The role of parallel computing software and hardware architectures will be researched in relation to managing large data sets and making both data mining algorithms and engineering design optimization models easier to solve. While traditional multilevel optimization algorithms have focused on efficiently solving decomposed

problems, this research will focus on proposing novel optimization techniques that can leverage the computational resources of distributed computing. The computational resources provided at the San Diego Super Computing center and the National Center for Supercomputing Applications (NCSA) will be used to address several of the key design problems (Example found in the Appendix). Researchers are provided with access to several Supercomputing resources including the IBM Power4 Data Star, the IA-64 Linux Cluster, OnDemand (Rocks-131 Cluster) and Thor. Each of these High Performance Computer systems can address a wide range of engineering design optimization problems and will be investigated in future research projects.

Chapter 8

References

[1] J. McEntire. *D2K Toolkit User Manual*. National Center for Supercomputing Applications (NCSA), Office of Technology Management 308 Ceramics Building, MC-243105 South Goodwin AvenueUrbana, Illinois 61801-2901, 1 edition, April 2003.

[2] Ian H. Witten and Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. *SIGMOD Rec.*, 31(1):76–77, 2002.

[3] S. K. Moon, S. R. T. Kumara, and T. W. Simpson. Data mining and fuzzy clustering to support product family design. In *Proceedings of DETC 06, 2006 ASME Design Engineering Technical Conferences*, number DETC2006/DAC-99287, Philadelphia, PA, September 2006.

[4] J. Jiao and Y. Zhang. Product portfolio identification based on association rule mining. *Computer-Aided Design*, pages 149–172, May 2004.

[5] T. W. Simpson. Product platform design and optimization: Status and promise. In *Proceedings of DETC 06, 2006 ASME Design Engineering Technical Conferences*, number DETC2003/DAC-48717, Chicago, IL, September 2003.

[6] S. Berry and A. Pakes. The pure characteristics demand model. *Journal of Economics*, December 2005.

[7] R. Farrell and T. Simpson. Product platform design to improve commonality in custom products. *Journal of Intelligent Manufacturing*, 14:541–556, 2003.

[8] R. Fellini, M. Kokkolaras, and P. Papalambros. Quantitative platform selection in optimal design of product families, with application to automotive engine design. *Journal of Engineering Design*, 17(5):429446, October 2006.

[9] S. Fixon. Product architecture assessment: a tool to link product, process, and supply chain design decisions. *Journal of Operations Management*, page 345369, November 2004.

[10] M. J. Scott, J. Arenillas, T. W. Simpson, S. Valliyappan, and V. Allada. Towards a suite of problems for comparison of product platform design methods: A proposed classification. In *Proceedings of DETC 06, 2006 ASME Design Engineering Technical Conferences*, number DETC2006/DAC-99289, Philadelphia, PA, September 2006.

[11] M.E. Pullmana, W.L. Mooreb, and D.G. Wardellb. A comparison of quality function deployment and conjoint analysis in new product design. *The Journal of Product Innovation Management*, 19(1):354–364, 2002.

[12] J.J. Cristiano, J.K. Liker, and C.W. White III. Customer-driven product development through quality function deployment in the u.s. and japan. *The Journal of Product Innovation Management*, 17(1):286–308, 2000.

[13] A. Lowe, K. Ridgway, and H. Atkinson. Qfd in new production technology evaluation. *Int. J. Production Economics*, 67:103–112, 2000.

[14] K. Ashihara and K. Ishii. Application of quality function deployment for new business r and d strategy development. In *2005 ASME International Mechanical Engineering Congress and Exposition*, Orlando, Florida USA, 2005.

[15] P.E. Green, A.M. Krieger, and Y.J. Wind. Thirty years of conjoint analysis:reflections and prospects. *Journal of Interfaces*, 31(3):56–73, May-June 2001.

[16] W.L. Moore, J.J. Louviere, and R. Verma. Using conjoint analysis to help design product platforms. *Journal of Innovation Management*, 16:27–39, 1999.

[17] M.D. Grissom, A.D. Belegundu, A. Rangaswamy, and G.H. Koopmann. Conjoint-analysis-based multiattribute optimization: application in acoustical design. *Struct Multidisciplinary Optimization*, 31:8–16, 2006.

[18] J. J. Michalek, F.M. Feinberg, and P.Y. Papalambros. Linking marketing and engineering product design decisions via analytical target cascading. *Journal of Product Innovation Management: Special Issue on Design and Marketing in New Product Development*, 22:42–62.

[19] A.T. Olewnik and K.E. Lewis. Conjoint-HOQ: A quantitative methodology for consumer-driven design. In *Proceedings of the ASME 2007 IDET Conferences and CIE Conference IDETC/CIE 2007*. ASME, Septermber 2007.

[20] H. Li and S. Azarm. Product design selection under uncertainty and with competitive advantage. *Transactions of ASME: Journal of Mechanical Design*, 122:411–418, December 2000.

[21] H. J. Wassenaar, W. Chen, J. Cheng, and A. Sudjianto. Enhancing discrete choice demand modeling for decision-based design. *Transactions of ASME: Journal of Mechanical Design*, 127:514–523, July 2005.

[22] S. T. Berry. Estimating discrete-choice models for product differentiation. *RAND Journal of Economics*, 25(2):242–262, Summer 1994.

[23] D. Kumar. *Demand Modeling for Enterprise-Driven Product Design*. PhD thesis, Northwestern University, December 2007.

[24] H. M. Kim, D. Kumar, and W. Chen. Target exploration for disconnected feasible regions in enterprise-driven multilevel product design. *AIAA Journal*, 44:67–77, 2006.

[25] C. S. Tucker and H. M. Kim. Product family decision tree concept generation and validation through data mining and multi-level optimization. In *Proceedings of the 33rd ASME Design Automation Conference*, Las Vegas, NV, September 2007. ASME.

[26] M. Ben-Akiva and S. R. Lerman. *Discrete Choice Analysis:Theory and Application to Travel Demand.* MIT Press, December 1985.

[27] D. Braha. *Data Mining for Design and Manufacturing*. Kluwer Academic Publishers, P.O. Box 17, 3300 AA Dordrecht, The Netherlands, 2001.

[28] A. Kusiak. Data mining: manufacturing and service applications. *International Journal of Production Research,*, 44(18-19):41754191, October 2006.

[29] Conrad S. Tucker and Harrison M. Kim. Data-driven decision tree classification for product portfolio design optimization. *Journal of Computing and Information Science in Engineering*, 9(4):041004, 2009.

[30] B. AGARD and A. KUSIAK. Data-mining-based methodology for the design of product families. *International Journal of Production Research*, 42(15):2955–2969, August 2004.

[31] J. Nanda, T. Simpson, S.R.T. Kumara, and S.B. Shooter. A methodology for product family ontology development using formal concept analysis and web ontology language. *Journal of Computing and Information Science in Engineering*, 6(103), 2006.

[32] K. F. Tsang, H. C. W. Lau, and S.K. Kwok. Development of a data mining system for continual process quality improvement. *Institution of Mechanical Engineers*, 221(179), October 2006.

[33] Chao-Ton Su, Yung-Hsin Chen, and D.Y. Sha. Linking innovative product development with customer knowledge: a data-mining approach. *Technovation*, 26(7):784 – 795, 2006.

[34] Mirko Böttcher, Martin Spott, and Rudolf Kruse. Predicting future decision trees from evolving data. In *Proceedings of ICDM '08*, pages 33–42, 2008.

[35] C. W. Günther, S. B. Rinderle, M. U. Reichert, and W. M. P. van der Aalst. Change mining in adaptive process management systems. In *Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS'06), Montpellier, France*, volume 4275 of *Lecture Notes in Computer Science*, pages 309–326, Berlin, Heidelberg, New York, November 2006. Springer Verlag.

[36] Peipei Li, Xuegang Hu, and Xindong Wu. Mining concept-drifting data streams with multiple semi-random decision trees. pages 733–740. 2008.

[37] J. G. Wacker and M. Trelevan. Component part standardization: An analysis of commonality sources and indices. *Journal of Operations Management*, 6(2):219–244, 1986.

[38] A. Messac, M. Martinez, and T.W. Simpson. Effective product family design using physical programming. *Engineering Optimization*, 34:245–261, 2002.

[39] Fabrice Alizon, Kiran Khadke, Henri J. Thevenot, John K. Gershenson, Tucker J. Marion, Steven B. Shooter, and Timothy W. Simpson. Frameworks for product family design and development. *Concurrent Engineering: R&A*, 15(2):187–199, 2007.

[40] Z. Dai and M.J. Scott. Effective product family design using preference aggregation. *Journal of Mechanical Design*, 128, July 2006.

[41] O. de Weck, E. Suh, and D. Chang. Product family and platform portfolio optimization. In *Proceedings of DETC03, 2003 ASME Design Engineering Technical Conferences*, number DETC03/DAC-48721, Chicago, IL, September 2003.

[42] J. P. Gonzales-Zugasti, K. N. Otto, and J. D. Baker. Assessing value in platformed product family design. *Research in Engineering Design*, 13(1):30–41, August 2001.

[43] P. Desai, S. Kekre, S. Radhakrishnan, and K. Srinivasan. Product differentiation and commonalityin design: Balancing revenue and cost drivers. *Management Science*, 47(1):37–51, January 2001.

[44] K. Kim and D. Chhajed. Commonality in product design: Cost saving, valuation change and cannibalization. *European Journal of Operational Research*, (125):602–621, 2000.

[45] Henri J. Thevenot and Timothy W. Simpson. Commonality indices for product family design: a detailed comparison. *Engineering Design*, 17(2):99–119, 2006.

[46] D. Collier. The measurement and operating benefits of component part commonality. *Decision Sciences*, 12(1):85–96, 1981.

[47] H. Thevenot and T. Simpson. A comparison of commonality indices for product family design. *Proceedings of DETC04 ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, September 28-October 2, 2004.

[48] J. Jiao and M. M. Tseng. Understanding product family for mass customization by developing commonality indices. *Journal of Engineering Design,*, 11(3):225–243, 2000.

[49] S. Kota, K. Sethuraman, and R. Miller. A metric for evaluating design commonality in product families. *ASME Journal of Mechanical Design*, 122(4):403–410, 2000.

[50] Y. Nomaguchi and K. Fujita. Product architecture design process for model-based knowledge management. *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN,ICED 05 MELBOURNE,*, August 15-18 2005.

[51] Z. Siddique, D. W. Rosen, and N. Wang. On the applicability of product variety design concepts to automotive platform commonality. *Design Theory and Methodology, Atlanta, GA, ASME, Paper No. DETC98/DTM-5661.*, September 13-16 1998.

[52] M. Martin and K. Ishii. Design for variety: A methodology for understanding the costs of product proliferation. *Design Theory and Methodology*, August 18-22 1996.

[53] H. Thevenot and T. Simpson. A comprehensive metric for evaluating component commonality in a product family. *Proceedings of DETC06 ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, September 10-13 2006.

[54] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *American Association for Artificial Intelligence*, 1996.

[55] C. S. Tucker and H. M. Kim. Optimal product portfolio formulation by merging predictive data mining with multilevel optimization. *Transactions of ASME: Journal of Mechanical Design*, 130:991–1000, 2008.

[56] J. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi. *Knowledge Discovery in Databases: PKDD 2004*. Springer, 2004.

[57] Thomas Haigh. "a veritable bucket of facts" origins of the data base management system. *SIGMOD Rec.*, 35:33–49, June 2006.

[58] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2 edition, 2008.

[59] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[60] Peter P. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[61] I. Bruha and F. Franek. Comparison of various routines for unknown attribute value processing: the covering paradigm. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(9):939–955, 1996.

[62] J. W. Grzymala-Busse and M. Hu. A comparison of several approaches to missing attribute values in data mining. *LNAI 2005*, pages 378–385, 2001.

[63] J.M. Ling, J.M. Aughenbaugh, and C.J. Paredis. Managing the collection of information under uncertainty using information economics. *Transactions of ASME: Journal of Mechanical Design*, 128(4):980–990, 2006.

[64] D.J. Hand. Data mining: Statistics and more? *The American Statistician*, 52(2):112–118, 1998.

[65] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, February 2006.

[66] M. Campos, P. Stengard, and B. Milenova. Data-centric automated data mining. *4th International Conference on Machine Learning and Applications*, 2005.

[67] K.F. Hulme and C.L. Bloebaum. A simulation-based comparison of multidisciplinary design optimization solution strategies using cascade. *Struct Multidisc Optim*, 19:17–35, 2000.

[68] C.D. McAllister, T.W Simpson, K. Hacker, K Lewis, and A. Messac. Integrating linear physical programming within collaborative optimization formultiobjectivemultidisciplinary design optimization. *Struct Multidisc Optim*, 29:178–189, 2005.

[69] R.D. Braun. *Collaborative optimization: an architecture for large-scale distributed design*. Ph.d. thesis, Stanford University, 1996.

[70] Sobieszczanski-Sobieski J., Agte J.S., and Sandusky Jr R.R. Bilevel integrated system synthesis. *AIAA*, 38(1):164–172, 2000.

[71] R.T. Haftka and L.T. Watson. Multidisciplinary design optimization with quasiseparable subsystems. *Optim Eng*, 6(1):9–20, 2005.

[72] H. M. Kim, D. G. Rideout, P. Y. Papalambros, and J. L. Stein. Analytical target cascading in automotive vehicle design. *Transactions of ASME: Journal of Mechanical Design*, 125(3):481–489, 2003.

[73] H. M. Kim, W. Chen, and M. M. Wiecek. Lagrangian dual coordination for analytical target cascading. *AIAA Journal*, 2006. In press. Also presented at 2005 IFORS Conference, Honolulu, HI.

[74] N. F. Michelena, H. Park, and P.Y. Papalambros. Convergence properties of analytical target cascading. volume 2, 2002. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA, AIAA-2002-5506.

[75] H.M. Kim, N.F. Michelena, P.Y. Papalambros, and T. Jiang. Target cascading in optimal system design. *Journal of Mechanical Design*, 123(1), June 2003.

[76] S. Tosserams, L.F.P. Etman, and J.E. Rooda. An augmented lagrangian decomposition method for quasi-separable problems in mdo. *Struct Multidisc Optim*, 2006.

[77] H.M. Kim, M. Kokkolaras, L.S. Louca, G.J. Delagrammatikas, N.F. Michelena, Z.S. Filipi, P.L. Papalambros, J.L. Stein, and D.N. Assanis. Target cascading in vehicle redesign: A class vi truck study. *International Journal of Vehicle Design*, 3(3):199–225, 2002.

[78] M. Kokkolaras, R. Fellini, H.M. Kim, N.F. Michelena, and P.Y. Papalambros. Extension of the target cascading formulation to the design of product families. *Struct Multidisc Optim*, 24:293–301, 2002.

[79] P. A. Flack and N. Lachiche. Naïve bayes classification of structued data. *Machine Learning*, 57(3):233–269, December 2004.

[80] I. Rish. An empirical study of the naive bayes classifier. *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, 2001.

[81] H. Zhang, C. Ling, and Z. Zhao. The learnability of naive bayes. *Canadian AI2000, LNAI 1822*, pages 432–441, 2000.

[82] D. Meretakis and B. Wuthrich. Extending naive bayes classifiers using long itemsets. *5th ACM SIGKDD International Conference on Knowledge Discovery and data mining*, pages 164–174, 1999.

[83] E. M. Kleinberg. *Lecture Notes in Computer Science*, volume 1857/2000. Springer Berlin/Heidelberg, 2000.

[84] M. Degroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.

[85] G. Sparacino, C. Tombolato, and C. Cobelli. Maximum-likelyhood versus maximum a posteriori parameter estimation of physiological system models: The c-peptide impulse response case study. *IEE Transactions on Biomedial Engineering*, 47(6), June 2000.

[86] K. Holmstrom, A. O. Goran, and M. M. Edvall. *USERS GUIDE FOR TOMLAB /MINLP*. Tomlab Optimization, Tomlab Optimization Inc, 855 Beech St 121 San Diego CA USA, February 2006.

[87] A. B. Cooper, P. Georgiopoulos, H. M. Kim, and P. Y. Papalambros. Analytical target setting: An enterprise context in optimal product design. *ASME*, 2006.

[88] D. Canback. Diseconomies of scale in large corpotations. Technical description, Canback Dangel Predictive Analytics Advisors, 10 Derne Street, Boston, MA 02114, 2003.

[89] H. M. Kim, N. F. Michelena, P. Y. Papalambros, and T. Jiang. Target cascading in optimal system design. *Transactions of ASME: Journal of Mechanical Design*, 125(3):474–480, 2003.

[90] CBW. Standby and talk times. Cell Phone Batteries, 2006.

[91] Y. Neuvo. Cellular phones as embedded systems. *IEEE International Solid-State Circuits Conference*, June 2004.

[92] The Math Works, Inc., Natick, MA. *Optimization Toolbox for Use with MATLAB*, version 2 edition, 1999.

[93] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[94] Igor Kononenko. Estimating attributes: analysis and extensions of relief. In *ECML-94: Proceedings of the European conference on machine learning on Machine Learning*, pages 171–182, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[95] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[96] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[97] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[98] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data (Prentice Hall Advanced Reference Series : Computer Science)*. Prentice Hall College Div, March 1988.

[99] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 106–113, New York, NY, USA, 1998. ACM.

[100] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, 2002.

[101] Thaddeus Tarpey. A parametric k-means algorithm. *Comput. Stat.*, 22(1):71–89, 2007.

[102] Robert E. Kass and Larry Wasserman. A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. *Journal of the American Statistical Association*, 90(431):928–934, 1995.

[103] H. M. Kim. *Target Cascading in Optimal System Design*. PhD thesis, The University of Michigan, Ann Arbor, MI, 2001.

[104] H. M. Kim, M. Kokkolaras, L. Louca, G. Delagrammatikas, N. F. Michelena, Z. Filipi, P. Y. Papalambros, and D. Assanis. Target cascading in vehicle redesign: A class VI truck study. *International Journal of Vehicle Design*, 29(3):199–225, 2002.

[105] M. Kokkolaras, L.S. Louca, G.J. Delagrammatikas, N.F. Michelena, Z.S. Filipi, P.Y. Papalambros, J.L. Stein, and D.N. Assanis. Simulation-based optimal design of heavy trucks by model-based decomposition: An extensive analytical target cascading case study. *International Journal of Heavy Vehicle Systems*, 11(3–4):402–432, 2004.

[106] J. Allison, D. Walsh, M. Kokkolaras, P. Y. Papalambros, and M. Cartmell. Analytical target cascading in aircraft design. Reno, Nevada, January 2006. AIAA-2006-1325. 44th AIAA Aerospace Sciences Meeting and Exhibit.

[107] N. M. Alexandrov and R. M Lewis. Analytical and computational aspects of collaborative optimization for multidisciplinary design. *AIAA Journal*, 40(2):301–309, 2002.

[108] Y. Zhang. *Indoor air quality engineering*. CRC Press, Boca Raton, FL, 2005.

[109] D. Barker. Development of an optimization design platform for aerodynamic particle separators. Ms, University of Illinois at Urbana-Champaign, 2008.

[110] J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[111] B. Hunt, J. Marin, and P. Stone. *Experiments in Induction*. Academic Press, New York, NY, 1966.

[112] N. Amor, S. Benferhat, and Z. Elouedi. Naive bayes vs decision trees in intrusion detection systems. *2004 ACM Symposium on Applied Computing*, 2004.

[113] J. Quinlan. *C4.5: Programs for machine learning*, volume 1. Morgan Kaufmann Publishers, 1993.

[114] S. Salzberg. Book review: C4.5:by j. ross quinlan.inc., 1993.programs for machine learning morgan kaufmann publishers,. *Machine Learning*, 16:235–240, 1994.

[115] J.W. Grzymala-Busse and J. Stefanowsk. Three discretization methods for rule induction. *International Journal of Intelligent Systems*, 16:29–38, 2001.

[116] P. Perner and S. Trautzsch. Multi-interval discretization methods for decision tree learning. *Advances in Pattern Recognition,*, pages 475–482, 1998.

[117] J. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[118] J. Quinlan. Learning with continuous classes. In *Proceedings of Artificial Intelligence (Adams and Sterling, Eds*, pages 343–348, 1992.

[119] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[120] I. J. Hidalgo and H. M. Kim. System of systems optimization by pseudo-hierarchical multistage model. In *Proceedings of the 11th AIAA/MAO Conference*, number AIAA-2006-6921, Portsmouth, VA, September 2006.

[121] C. Tucker and H.M. Kim. Cell phone customer survey: Online interactive user interface created using webtools. https://webtools.uiuc.edu/survey/Secure?id=5617516 (accessed October, 2006).

[122] C.S. Tucker, H.M. Kim, D.E. Barker, and Y. Zhang. A relieff attribute weighting and x-means clustering methodology for top-down product family optimization. *Engineering Optimization*, 42(7):593–616, September 2010.

[123] I.Guyon B. Boser and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[124] H. Zhang. Classification trees with multiple binary responses. *Journal of the American Statistical Association*, 93(441):180–193, 98.

[125] R. Siciliano and F. Mola. Multivariate data analysis and modeling through classification and regression trees. *Computational Statistics and Data Analysis*, 32:285–301, 2000.

[126] S. Kim and K. B. Lee. Constructing decision trees with multiple response variables. *International Journal of Management and Decision Making*, 4(4):337–353, 2003.

[127] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[128] Tianyi Wu, Yuguo Chen, and Jiawei Han. Association mining in large databases: A re-examination of its measures. pages 621–628. 2007.

[129] et al. Wets, G. Identifying decision structures underlying activity patterns: An exploration of data mining algorithms. *Transportation Research Record*, 1718:1–9, 2000.

[130] L. J. Xie, C. and E. Parkany. Work travel mode choice modeling with data mining decision trees and neural networks. *Journal of the Transportation Research Board*, 1854, 2003.

[131] R. Kitamura Yamamoto, T. and J. Fujii. Drivers' route choice behavior: Analysis by data mining algorithms. *Journal of the Transportation Research Board*, 1807:59–66, 2002.

[132] S. Kurauchi Yamamoto, Y. and T. Morikawa. Comparison of non-compensatory models of driver's choice on dynamic park and ride. in intelligent transportation systems. In *9th World Congress Proceedings*, 2002.

[133] A. Mohammadian and E.J. Miller. Nested logit models and artificial neural networks for predicting household automobile choices: Comparison of performance. *Journal of the Transportation Research Board*, 1807:92–100, 2002.

[134] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[135] Geraldine E. Rosario, Elke A. Rundensteiner, David C. Brown, and Matthew O. Ward. Mapping nominal values to numbers for effective visualization. In *in Special issue of selected and extended InfoVis 03 paper*, 2004.

[136] Wai Lam, Chi-Kin Keung, and Charles X. Ling. Learning good prototypes for classification using filtering and abstraction of instances. *Pattern Recognition*, 35:1491–1506, 2001.

[137] Yong Wang. A new approach to fitting linear models in high dimensional spaces. Technical report, University of Waikato: Waikato, 2000.

[138] Jianzhao Shen and Sujuan Gao. A Solution to Separation and Multicollinearity in Multiple Logistic Regression. *Journal of data science : JDS*, 6(4):515–531, October 2008.

[139] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms. *Mach. Learn.*, 40(3):203–228, September 2000.

[140] S. B. Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, 31(3), 2007.

[141] C.S. Tucker, C. Hoyle, H.M. Kim, and W. Chen. A comparative study of data-intensive demand modeling techniques in relation to product design and development. In *Proceedings of the ASME Design Engineering Technical Conferences*, San Diego, CA, 2009. DETC2009-87049.

[142] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006.

[143] C. Chatfield. The holt-winters forecasting procedure. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 27(3):264–279, 1978.

[144] H. Grubb and A. Mason. Long lead-time forecasting of uk air passengers by holt-winters methods with damped trend. *International Journal of Forecasting*, 17(1):71 – 82, 2001.

[145] Sheng Yue, Paul Pilon, and George Cavadias. Power of the mann-kendall and spearman's rho tests for detecting monotonic trends in hydrological series. *Journal of Hydrology*, 259(1-4):254 – 271, 2002.

[146] R. B. Cleveland, W. S. Cleveland, J. E. Mcrae, and I. Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[147] B.L. Smith, B.M. Williams, and R.K. Oswald. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, 10(4):303 – 321, 2002.

[148] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[149] Zheng Zhao and Huan Liu. Searching for interacting features in subset selection. *Intell. Data Anal.*, 13(2):207–228, 2009.

[150] Henry B. Mann. Nonparametric tests against trend. *Econometrica*, 13(3):245–259, 1945.

[151] Maurice Kendall and Jean D. Gibbons. *Rank Correlation Methods*. A Charles Griffin Title, 5 edition, September 1990.

[152] H.J. Wassenaar, W. Chen, J. Cheng, and A. Sudjianto. Enhancing discrete choice demand modeling for decision-based design. *Journal of Mechanical Design*, 127(4):514–523, 2005.

[153] K.E. Lewis, W. Chen, and L.C. Schmidt, editors. *Decision Making in Engineering Design*. ASME, 2006.

[154] J. *Demšar*. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7(1):1–30, 2006.

[155] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimiality of the simple bayesian classifier. In *Machine Learning*, pages 105–112, 1996.

[156] Chrysanthos Dellarocas. The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Management Science*, 49(10):1407–1424, 2003.

[157] Y. Chen and J. Xie. Online consumer review: Word-of-mouth as a new element of marketing communication mix. *Management Science*, 54(3):477–491, 2008.

[158] Jeremy Ginsberg, Matthew H. Mohebbi, Rajan S. Patel, Lynnette Brammer, Mark S. Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, November 2008.

[159] T. Ferguson. Online patient-helpers and physicians working together: a new partnership for high quality health care. *BMJ Journal*, 321(7269):1129–1132, November 2000.

[160] H. J. Wassenaar and W. Chen. An approach to decision-based design with discrete choice analysis for demand modeling. *Transactions of ASME: Journal of Mechanical Design*, 125:490–497, September 2003.

[161] P. Chandon, V.G. Morwitz, and W.J. Reinartz. Do intentions really predict behavior? self-generated validity effects in survey research. *Journal of Marketing*, 69(2):1–14, 2005.

[162] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 519–528, New York, NY, USA, 2003. ACM.

[163] Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

[164] Janyce Wiebe. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 735–740. AAAI Press / The MIT Press, 2000.

[165] N. Jindal and B. Liu. Mining comparative sentences and relations. In *Proceedings of AAAI*, 2006.

[166] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA, 2004. ACM.

[167] Xinghua Hu and Bin Wu. Classification and summarization of pros and cons for customer reviews. In *WI-IAT '09: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 73–76, Washington, DC, USA, 2009. IEEE Computer Society.

[168] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[169] J. Pei and J. Han. Constrained frequent pattern mining: a pattern-growth view. *SIGKDD Explor. Newsl.*, 4(1):31–39, 2002.

[170] Sebastian Celis and David R. Musicant. Weka-parallel: Machine learning in parallel. Technical report, Carleton College, CS TR, 2002.

[171] J. Quinlan. *C4.5 Programming for Machine Learning*. Morgan Kauffman, 1993.

[172] D. Parallel Shasha. C4.5 (pc4.5). cited 2008; Available from: http://www.cs.nyu.edu/ binli/pc4.5/., 1997.

[173] W. Liao. Parallel k-means data clustering. cited 2008; Available from: http://www.ece.northwestern.edu/ wkliao/Kmeans/index.html, 2005.

[174] S. Muller and H. Waller. Efficient integration of real-time hardware and web based services into matlab. In *ESS: 11th European Simulaiton Symposium and Exhibition*. Erlangen-Nuremberg, October 1999.

[175] Inc. Interactive Supercomputing. Star-p programming guide for use with matlab release 2.6.2, 2008.

[176] Mathworks. Matlab parallel computing toolbox. cited 2008; http://www.mathworks.com/products/parallel-computing, 2008.

[177] ILOG. Parallel cplex. cited 2008; Available from: http://www.ilog.com/products/cplex/product/parallel.cfm., 2008.

Chapter 9

Appendix

9.1 Cyberinfrastructure in Product Design

The Data Driven Product Design Platform is a user friendly interface that integrates customer knowledge discovery with engineering design. The Java based GUI was designed using the Eclipse Java Development Toolkit and the Jigloo extension of Eclipse for GUI based code development. The overall layout of the GUI is presented in Figure 9.1. The user can interact with both phases of the product design process. That is, 1) The Data mining Knowledge Discovery process and 2) The Engineering Design Simulation process. The details of each phase will be presented next and the benefits of high performance computing will be revealed for each of the phases.

KDD Toolkit: The KDD Toolkit allows the user to specify which data mining approach to utilize in the knowledge discovery process. The KDD Toolkit is partitioned into ***Parallel Data mining*** and ***Serial Data Mining*** as represented by Figure 9.2. Depending on the HPC resources available, parallel data mining can be used to reduce the computational time required to generate a set of product design targets. The parallel data mining applications that have been investigated thus far are 1). Parallel Weka, Parallel C4.5 and Parallel *K-means*.

1. **Parallel Weka** [170]: Is an extension of the Weka Open source environment. Parallel Weka parallelizes the n-fold cross validation stage of several well known classifications algorithms such as the J48 (Weka's implementation of C4.5 by Quinlan [171]). Given a data training set T, individual models can be built by partitioning the raw data set T into individual training sets and then later validated using the test set. Since each fold is independent of each other, this can be considered a *task parallel process*. Therefore, if there are n-folds, each taking t minutes, the parallelization theoretically reduces the computational complexity from n*t minutes to t*minutes. Actual results may be slightly below linear speedup due to the time lost as a result of message passing between processors.

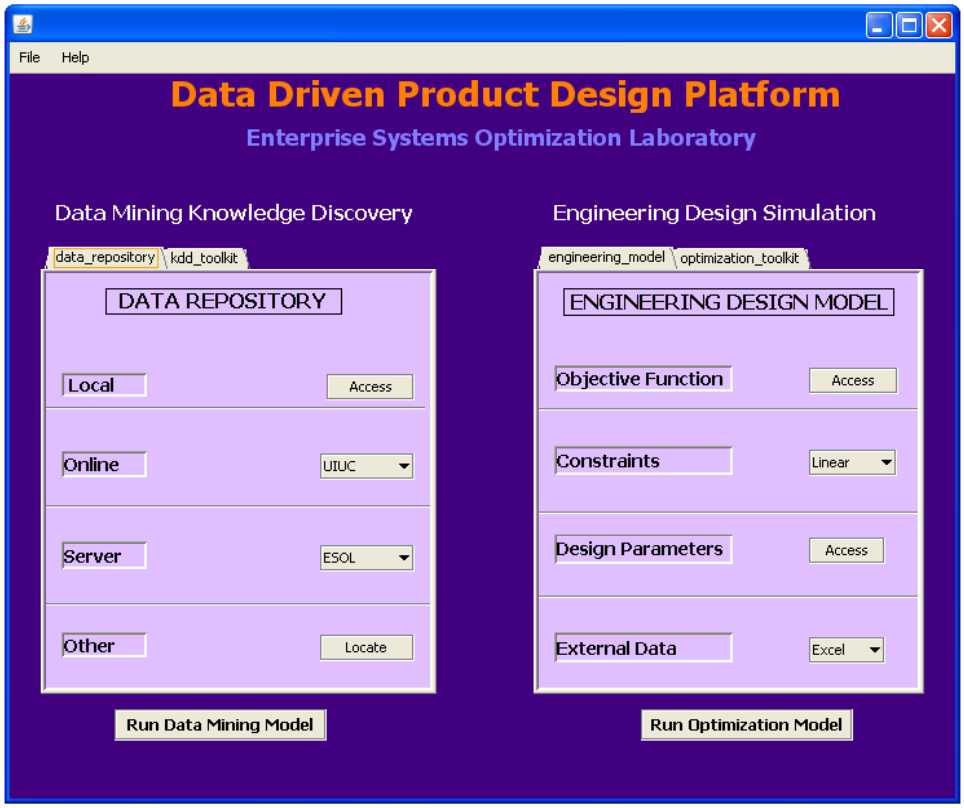


Figure 9.1: Java based Graphical User Interface (GUI) of Data Driven Product Design Platform

Weka Parallel Initialization:

1. (a) **Login to SDSC Cluster:** *thor.sdsc.edu*

   (b) **Set class path:** *java –cp weak.jar weak.core.Distributed Server 1234*

   (c) **Specify Weka Algorithm:** *weka.classifiers.j48.J48 –t weather.arff -a*

1. **Parallel C4.5** [172]: Parallel C4.5 is a classification based parallelization algorithm. The C4.5 is a decision tree based induction algorithm based on iteratively partitioning the data set with each attribute test that maximizes the information gain. PC4.5 runs on SunOS and Linux machines and is based on the Plinda distributed parallel computing system developed at New York University [172].

Weka Parallel Initialization:

1. (a) **Login to SDSC Cluster:** *tg-login*

   (b) **Unzip PC4.5:** *gunzip.pc4.5.tar then enter tar xvfpc4.5.tar*

   (c) **Compile PLinda using the make file** *make –f Linux*

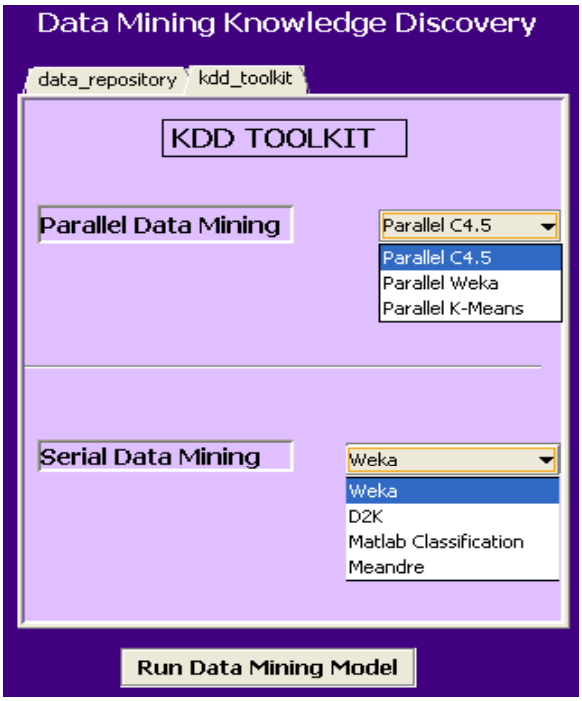


Figure 9.2: KDD Toolkit options allowing the user to choose between parallel and serial data mining

(d) **Note: Currently, there is a compatibility issue with a missing .h file "pix.h" that is preventing Plinda from successfully compiling on the tera-grid machines. A solution to this is currently being investigated.**

1. **Parallel K-Means** [173]: The K-means clustering algorithm is an unsupervised learning data mining algorithm. Unlike supervised learning data mining algorithms such as classification based C4.5 decision trees, Naïve Bayes , etc., the K-means algorithm determines patterns among the data set rather than predicting a class (performance variable). The goal is to parallelize the distance calculations at each iteration so as to minimize the time spent on determining the newly defined cluster centroid. Although this particular algorithm was not the focus of the research here, the final version of the Data Driven Product Design Platform will include a fully functional K-means option.

Parallel K-means Initialization:

1. (a) **Login to SDSC Cluster:** *dspoe.sdsc.edu*

   (b) **Save file to home directory**

   (c) **Compile MPI version:** *make mpi_main*

   (d) **Run:** *open_main.h*

(e) **Note: The initial code was run using SDSC DataStar, however MPI compatibility issues are currently being investigated before any large scale use of this algorithm will be available through the Data Driven Product Design Platform.**

Serial Data Mining

The Data Driven Product Design Platform will enable the user to access serial versions of popular open source as well as closed source data mining algorithms such as Weka, D2K, Matlab Classification Toolbox and the newly developed Meandre by NSCA. Due to the focus of the CIEG program as it relates to high performance computing, the explanation of the serial data mining programs will be minimized.

Phase 2: Engineering Design Simulation The DDPD provides users with a centralize location for the entire product portfolio development process where the Matlab computational engine will be accessed *behind the scenes* directly from the DDPD user interface. This is achieved by calling several Java routines, one being the JMatLink [174] that creates an open channel for Matlab commands to be executed directly from the DDPD.

A snapshot of the Java based code is as follows:

```
private AbstractAction getRun_optmodel() {
    if(run_optmodel == null) {
    run_optmodel = new AbstractAction("Run", null) { //access action "Run" button
    public void actionPerformed(ActionEvent evt) {
    JMatLink engine = new JMatLink(); //Define "engine" from JMatLink
    System.out.println("Linking Data Knowledge Discovery With Engineering Design");
    System.out.println(" ");
    engine.engOpen(); //function to open the Matlab computational engine
    engine.engOutputBuffer();//initialization of output buffer
BufferedReader in = new BufferedReader( new InputStreamReader(System.in));
    String input = "";
    input = jTextField1.getText(); //input from the user "ATC_example"
    if (!input.equals("end") && !input.equals("exit")) //test for termination
    engine.engEvalString(input); //evaluates the string that the user inputs
    System.out.println(engine.engGetOutputBuffer());
    engine.engClose(); //end process
    }
    };
```

}

return run_optmodel;

}

Here, the user will input the exact name of the script that will call the Matlab Computational Engine. In this example, assuming that the script name is "ATC_example", the Matlab computational engine will then be accessed (engine.engOpen()) and the optimization sequence will begin. (**Note:** *In order for Matlab to recognize the script, the folder which contains the script must be in the Matlab command path*). If the user specifies the results file of the Data Mining Knowledge Discovery Process, the Matlab optimization model can directly read from this data file and hence avoid manual input of each product design target.

Optimization ToolKit:

The optimization toolkit interface allows the user to specify which optimization approach is most suitable for the specific design problem. Depending on the number of sequential processes, the user may choose to solve the problem using parallel data mining techniques. In the event where there are dependencies among different tasks or licensing restrictions, parallel optimization may not be feasible and the user also has the option of evoking serial optimization techniques.

Parallel Optimization: The 3 parallel optimization approaches investigated thus far are the StarP [175], Parallel Matlab [176] and Parallel CPLEX [177]. The parallel optimization approach that is utilized in the product design process relies heavily on the type of optimization problem being solved. The example engineering design problem involves both task parallelization and data parallelization. Research efforts will initially focus on developing model parallelization using the StarP client and then branching out to include C/C++ and Fortran based design models that utilize parallel CPLEX.

Task Parallelization:

The results from the data mining knowledge discovery process generate an ***m x n*** matrix of product design specifications, where ***m*** represents the number of unique attribute combinations and ***n*** represents the values for each attribute. As the size of the decision tree model increases, it often becomes computationally impractical for all product design candidates to be investigated hereby limiting the effectiveness of the product design process. Task Parallelization using StarP client aims to address some of these challenges facing the product portfolio design process by enabling a ***broadcast*** of independent tasks to different processors. We first begin by introducing the general form of the a specific type of optimization problem and how it relates to data parallelization.

Data Parallelization:

Mixed Integer Nonlinear Program General Formulation:

minimize f(x, y)

(Nonlinear objective function)

subject to g(x, y) = 0, h(x,y)=0 (Nonlinear Constraints)

Ax=b (Linear Constraints)

x ł X (Continuous variable bounded by X)

y ł Y integer (Integer Variables bounded by Y)

Since many engineering design models are mixed integer nonlinear programming problems, the goal is to parallelize each node of the branch (Gradient Information), Where S= Master Problem and Si=subproblem (i). Each S_i has optimal solution. Therefore S*=min(S_1*, S_2*,... S_k*), where k=number of sub problems. Currently, StarP has the ability to parallelize nonlinear constraint optimization functions that use the in-built Matlab function ***fmincon***. The StarP equivalent to fmincon is ***ppfmincon*** and takes the form:

[x y] = ppfmincon(@sys_fun, x0, extra_args)

Where sys_fun is the constrained nonlinear optimization problem to be solved with starting variable values at x0 and output x and y. ***ppfmincon*** parallelizes the gradient and function evaluations of a constrained nonlinear programming problem.

The StarP user interface allows users to specify the number of processors required to solve their optimization problem. In the case of the task parallel problem, we will set the number of processors *n* equal to the number of leaves generated by the decision tree classification algorithm so that each optimization problem can run simultaneously on the OnDemand cluster. If parallelization of the actual optimization algorithm is also occurring (fmincon), then the number of processor may be altered to achieve maximum scale-up. The StarP Task Parallel function call is as follows: [out1 out2 ... outN] = **ppeval**('ATC_example', In1, In2, ..., InN);

Where outi and Ini are the outputs and inputs respectively to the *user/matlab function (ATC_example)*

The level of code customization depends on the overall objective of the design problem. For example one may choose to split or broadcast the inputs of the function using different StarP commands such as ppevalsplit or ppbcast.

Each task parallel problem is modeled based on the Analytical Target Cascading (ATC) decomposition approach.

The two-level model represents interaction with a system level that is system level is responsible for acquiring the results from the data mining approach (vector of design targets) and a lower subsystem level that attempts to match the higher level targets.

Evaluating Computational Time Savings (Serial VS Parallel)

The computational time savings between serial and parallel versions of the optimization model are highly dependent on the type of problem being solved and the structure of the model. To evaluate differences in run time, simple benchmarking commands can be used with the in-built Matlab functions *tic* and *toc* for the serial version and *pptic* and *pptoc* for the StarP parallel version. An example of the parallel time evaluation wrapped around a constraint optimization function call is given below (This is for the ATC_example presented earlier):

```
pptic;
x_sys= ppfmincon(@f_sys, x_sys, [], [], [], [], vlb, vub, @NONLCON_sys, options);
pptoc;
```

The output from the Matlab command window allows the user to see how many messages were passed between the client and server machines as well as the total time required to accomplish this. This vital piece of information can then be used to benchmark the computational time savings with serial versions of the code.

```
Client/server communication report:
Sent by server: 162 messages, 1.263e+004 bytes
Received by server: 162 messages, 2.073e+004 bytes
Total communication time: 6.991e-003 seconds
Server processing report:
Duration of calculation on server (wall clock time): 3.330e+000s
#ppchangedist calls: 0
-------------------------------------------------------
Total time: 2.972e+001 seconds
```

Figure 9.3: Enterprise Objective Function

7/12/07 1:07 PM C:\MATLAB\work\ASME_matlab\ASME_DAC1\Optimal Prod...\ent_FUN2_1.m 1 of 1

```
function [f_sys] = ent_FUN(ent_SYS2_1,Prob)

global response_SUBSYS;

%Stage 1: SMS Architecture Design


TFT=1;
T_Battery=3;
profit=ent_SYS2_1(1);
T_Display=TFT;

E_Battery=ent_SYS2_1(12);
E_Connection1=ent_SYS2_1(13); %Bluetooth
E_Connection2=ent_SYS2_1(14); %Wifi
E_Connection3=ent_SYS2_1(15); %Infrared
E_Design1=ent_SYS2_1(16); %Shell Phone
E_Design2=ent_SYS2_1(17); %Flip Phone
E_Display1=ent_SYS2_1(18); %TFT LCD
E_Display2=ent_SYS2_1(19); %OLED LCD

E_Cost=ent_SYS2_1(11);

f_sys = -profit+ 1e5*((T_Battery-ent_SYS2_1(3))^2+(T_Display-ent_SYS2_1(9))^2) ...
+1e3*E_Battery +1e3*E_Connection1+1e3*E_Connection2+1e3*E_Connection3...
+1e3*E_Design1+1e3*E_Design2+1e3*E_Display1+1e3*E_Display2+1e5*E_Cost;
```

Figure 9.4: Enterprise NONLINEAR Constraints

7/12/07 1:08 PM C:\MATLAB\work\ASME_matlab\ASME_DAC1\Optimal ...\ent_NONLCON2_1.m 1 of 1

```
function constr = NONLCON_sys(ent_SYS2_1,Prob)
load SMS_SUB2_1.mat SMS_SUB2_1 -ascii ;
global response_SUBSYS;

%Customer information predicted by Data Mining C4.5 DT
Demand=907;
MaxPrice=160;
Cost=SMS_SUB2_1(87);
profit=ent_SYS2_1(1);

%Stage 1: SMS Architecture Design


% Tolerance constraint
g(1) = 1e0*(( ent_SYS2_1(2) - response_SUBSYS(1))^2) - ent_SYS2_1(11); %Cost
g(2) = 1e0*(( ent_SYS2_1(3) - response_SUBSYS(2))^2) - ent_SYS2_1(12)  ;%Talk Time
g(3) = 1e0*(( ent_SYS2_1(4) - response_SUBSYS(3))^2) - ent_SYS2_1(13) ;%Bluetooth
g(4) = 1e0*(( ent_SYS2_1(5) - response_SUBSYS(4))^2) - ent_SYS2_1(14);%Wifi
g(5) = 1e0*(( ent_SYS2_1(6) - response_SUBSYS(5))^2) - ent_SYS2_1(15);%Infrared
g(6) = 1e0*(( ent_SYS2_1(7) - response_SUBSYS(6))^2) - ent_SYS2_1(16);%Shell Phone
g(7) = 1e0*(( ent_SYS2_1(8) - response_SUBSYS(7))^2) - ent_SYS2_1(17) ;%Flip Phone
g(8) = 1e0*(( ent_SYS2_1(9) - response_SUBSYS(8))^2) - ent_SYS2_1(18);%TFT LCD
g(9) = 1e0*(( ent_SYS2_1(10) - response_SUBSYS(9))^2) - ent_SYS2_1(19) ;%OLED


h(1)=profit-Demand*(MaxPrice-Cost);

C=[g(1) g(2) g(3) g(4) g(5) g(6) g(7) g(8) g(9)]';

Ceq=[h(1)]';

constr = [C;Ceq];
```

Figure 9.5: Enterprise variable bound definitions

```
7/12/07 1:09 PM  C:\MATLAB\work\ASME_matlab\ASME_DAC1\Optimal Pro...\ent_OSYS2_1.m   1 of 2

% Cell Phone Design
% Target Cascading approach
% Two level problem: System-Subsystem

% System level problem


global response_SUBSYS, count;
% A=[]'; B=[]'; Aeq=[]'; Beq=[]'; % matrix/vectors for defining linear constraints (not↙
used)

load X_ALL2_1.mat X_ALL2_1 -ascii ;
load SMS_SUB2_1.mat SMS_SUB2_1 -ascii ;

ent_SYS2_1 =[ 200 X_ALL2_1(87) X_ALL2_1(45) X_ALL2_1(33) X_ALL2_1(47) X_ALL2_1(48)↙
X_ALL2_1(19) X_ALL2_1(25) X_ALL2_1(59) X_ALL2_1(66) 10 10 10 10 10 10 10 10]';

response_SUBSYS(1) = SMS_SUB2_1(87) ;  % Actually, response from subsystem---Cost
response_SUBSYS(2) = SMS_SUB2_1(45) ;  % Actually, response from subsystem---Talk Time
response_SUBSYS(3) = SMS_SUB2_1(46) ;  % Actually, response from subsystem---Bluetooth
response_SUBSYS(4) = SMS_SUB2_1(47) ;  % Actually, response from subsystem---Wifi
response_SUBSYS(5) = SMS_SUB2_1(48) ;  % Actually, response from subsystem---Infrared
response_SUBSYS(6) = SMS_SUB2_1(19) ;  % Actually, response from subsystem---Shell Phone
response_SUBSYS(7) = SMS_SUB2_1(25) ;  % Actually, response from subsystem---Flip Phone
response_SUBSYS(8) = SMS_SUB2_1(59) ;  % Actually, response from subsystem---TFT LCD
response_SUBSYS(9) = SMS_SUB2_1(66) ;  % Actually, response from subsystem---OLED


xlb=[-1000000000 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0]' ;
xub=[1000000000 300 7 1 1 1 1 1 1 1 10 10 10 10 10 10 10 10]';
xstatus=[0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]';


 c_L = [-inf -inf -inf -inf -inf -inf -inf -inf -inf 0 ]';
 c_U = [ 0 0 0 0 0 0 0 0 0 0]';
x_opt=[];

ConsPattern = [];         % All elements of grad c(x) are nonzero
HessPattern = zeros(1,1); % Linear obj.function => zero Hessian
VarWeight = [];
fIP   = [];       % An upper bound on the IP value wanted.
xIP   = [];       % Makes it possible to cut branches


Prob = minlpAssign('ent_FUN2_1', [], [], [], xlb, xub, 'SMS Phone_1 Profit Calculation',↙
ent_SYS2_1, ...
                        xstatus, VarWeight, [], [], ...
                        [], [], [], 'ent_NONLCON2_1', [], [], ConsPattern, c_L, c_U,↙
x_opt);
```

Figure 9.6: Script file calling the optimization sequence

```
7/12/07 1:09 PM  C:\MATLAB\work\ASME_matlab\ASME_DAC1\Optimal Pr...\overall_s2_1.m  1 of 2

% Cell Phone Product Architecture
% Target Cascading approach
% Three level problem: System-Subsystem-Component

% Overall Coordination Problem

clear all;
warning off;

global count ;



xm(1)=1;  xm(11)=1;  xm(21)=50;  xm(31)=1;   xm(40)=50;    xm(50)=1;     xm(60)=40;↙
xm(70)=18;           xm(80)=0;
xm(2)=0;  xm(12)=0;   xm(22)=30;  xm(32)=20;   xm(41)=35;    xm(51)=0;    xm(61)=30;↙
xm(71)= 11;          xm(81)=0;
xm(3)=0;  xm(13)=0;   xm(23)=40;   xm(33)=50;   xm(42)=11;     xm(52)=0;    xm(62)=25000;↙
xm(72)= 15;          xm(82)=0;
xm(4)=0;   xm(14)=0;  xm(24)=20;   xm(34)=40;   xm(43)=35;     xm(53)=0;    xm(63)=18;↙
xm(73)=0;           xm(83)=0;
xm(5)=0;  xm(15)=0;  xm(25)=0;   xm(35)=12;   xm(44)=1100;     xm(54)=0;    xm(64)=12;↙
xm(74)=0;           xm(84)=0;
xm(6)=0;  xm(16)=0;  xm(26)=70;   xm(36)=30;   xm(45)=5;     xm(55)=0;      xm(65)=9;↙
xm(75)=0;           xm(85)=0;
xm(7)=0;  xm(17)=0;   xm(27)=40;  xm(37)=1200;   xm(46)=0;    xm(56)=0;      xm(66)=0;↙
xm(76)=0;           xm(86)=0;
xm(8)=0; xm(18)=1;   xm(28)=25;    xm(38)=0;      xm(47)=1;    xm(57)=0;       xm(67) =35;↙
xm(77)=0;           xm(87)=40;
xm(9)=0; xm(19)=0;   xm(29)=20;    xm(39)=15;    xm(48)=0;     xm(58)=0;       xm(68) =25;↙
xm(78)=0;           xm(88)=60;
xm(10)=0; xm(20)=90;  xm(30)=15;                 xm(49)=0;    xm(59)=1;       xm(69)↙
=18000;  xm(79)=0;

X_ALL2_1=[xm(1:88)]';

xms(1)=1;  xms(11)=1;  xms(21)=50;  xms(31)=1;   xms(40)=50;    xms(50)=1;     xms(60)=40;↙
xms(70)=18;           xms(80)=0;
xms(2)=0;  xms(12)=0;   xms(22)=30;  xms(32)=20;   xms(41)=35;    xms(51)=0;    xms(61)↙
=30;         xms(71)= 11;          xms(81)=0;
xms(3)=0;  xms(13)=0;   xms(23)=40;   xms(33)=50;   xms(42)=11;     xms(52)=0;    xms(62)↙
=25000;    xms(72)= 15;          xms(82)=0;
xms(4)=0;   xms(14)=0;  xms(24)=20;   xms(34)=40;   xms(43)=35;     xms(53)=0;    xms(63)↙
=18;       xms(73)=0;           xms(83)=0;
xms(5)=0;  xms(15)=0;  xms(25)=0;   xms(35)=12;   xms(44)=1100;     xms(54)=0;    xms(64)↙
=12;       xms(74)=0;           xms(84)=0;
xms(6)=0;  xms(16)=0;  xms(26)=70;   xms(36)=30;   xms(45)=5;     xms(55)=0;      xms(65)↙
=9;        xms(75)=0;           xms(85)=0;
xms(7)=0;  xms(17)=0;   xms(27)=40;  xms(37)=1200;   xms(46)=0;    xms(56)=0;      xms(66)↙
=0;       xms(76)=0;           xms(86)=0;
xms(8)=0; xms(18)=1;   xms(28)=25;    xms(38)=0;      xms(47)=1;    xms(57)=0;       xms↙
(67) =35;   xms(77)=0;           xms(87)=40;
```

Figure 9.7: Engineering Objective Function

7/12/07 1:10 PM C:\MATLAB\work\ASME_matlab\ASME_DAC1\Optimal Prod...\SMS_FUN2_1.m 1 of 1

```
function [f]=SMS_FUN(SMS_SUB,Prob)

global target_SUB;

%The objective fucntion of Matching System Level Target with response

Cost=SMS_SUB(87);
Weight=SMS_SUB(88);
R_Battery=SMS_SUB(45);
R_Connection=SMS_SUB(46);
R_Display=SMS_SUB(59);
% R_Design=SMS_SUB(19);

f=Cost+ 1e3*((target_SUB(2)-R_Battery)^2+(target_SUB(8)-R_Display)^2);
```

Figure 9.8: Engineering Design Constraints


```
 function constr=NONLCON(SMS_SUB)
global target_SUB;
%University of Illinois-Urbana, Champaign
%Enterprise Systems Optimization Laboratory
%Dr. Harrison Kim: ISIE Assistant Professor
%Conrad Tucker   : ISIE Graduate Research Student


%Cell Phone Architecture Formulation
%************************Variable Assignments**************************


%-----------------------STORAGE AND MOMORY ASSIGNMENT--------------%

% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^RAM ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~32 MB RAM~~~~~~~~~~~~~~~~~~~~~~~~~
% MANUFACTURER 1: 32 Megabyte RAM
 % 32 MB Capacity
Mbyte_32_1=SMS_SUB(1); %RAM Discrete Variable
Weight_32MB1=1.1; % 32 MB Component weight
Cost_32MB1=0.2; % Unit cost of 32 MB RAM
Power_32MB1=2;  % Power Consumption per hour (Milliamps)

 % MANUFACTURER 2: 32 Megabyte RAM
 % 32 MB Capacity
 Mbyte_32_2=SMS_SUB(2); %RAM Discrete Variable
Weight_32MB2=2; % 32 MB Component weight
Cost_32MB2=0.35; % Unit cost of 32 MB RAM
Power_32MB2=2;  % Power Consumption per hour (Milliamps)

 % MANUFACTURER 3: 32 Megabyte RAM
 % 32 MB Capacity
 Mbyte_32_3=SMS_SUB(3); %RAM Discrete Variable
Weight_32MB3=1.3; % 32 MB Component weight
Cost_32MB3=0.15; % Unit cost of 32 MB RAM
Power_32MB3=3;  % Power Consumption per hour (Milliamps)

%---------------------------64 MB RAM----------------------------------

%MANUFACTURER 1: 64 Megabyte RAM
% 64 MB Capacity
  Mbyte_64_1=SMS_SUB(4); %RAM Discrete Variable
Weight_64MB1=1.2; % 32 MB Component weight
Cost_64MB1=0.43; % Unit cost of 32 MB RAM
Power_64MB1=4;  % Power Consumption per hour (Milliamps)

%MANUFACTURER 2: 64 Megabyte RAM
% 64 MB Capacity
 Mbyte_64_2=SMS_SUB(5); %RAM Discrete Variable
Weight_64MB2=2.5; % 32 MB Component weight
Cost_64MB2=0.51; % Unit cost of 32 MB RAM
```

Figure 9.8: Engineering Design Constraints (cont.)


```
Power_64MB2=1.5;  % Power Consumption per hour (Milliamps)

%MANUFACTURER 3: 64 Megabyte RAM
% 64 MB Capacity
 Mbyte_64_3=SMS_SUB(6); %RAM Discrete Variable
Weight_64MB3=2.3; % 32 MB Component weight
Cost_64MB3=0.41; % Unit cost of 32 MB RAM
Power_64MB3=3;  % Power Consumption per hour (Milliamps)
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%^^^^^^^^^^^^^^^^^^^^EXTERNAL STORAGE DEVICE^^^^^^^^^^^^^^^^^^^^^^^^^^

%-------------------------MS ProFlash Drive-----------------------------

%MANUFACTURER 1: MS Pro Flash Drive
MSP1=SMS_SUB(7); %EXTERNAL STORAGE Discrete Variable
Weight_MSP1=3.1;
Cost_MSP1=1.1; % Manufacturing cost of MS Pro slot
Power_MSP1=1;  % Power Consumption per hour (Milliamps)

%MANUFACTURER 2: MS Pro Flash Drive
MSP2=SMS_SUB(8); %EXTERNAL STORAGE Discrete Variable
Weight_MSP2=3.2;
Cost_MSP2=1.3; % Manufacturing cost of MS Pro slot
Power_MSP2=1.1;  % Power Consumption per hour (Milliamps)

%MANUFACTURER 3: MS Pro Flash Drive
MSP3=SMS_SUB(9);%EXTERNAL STORAGE Discrete Variable
Weight_MSP3=2.5;
Cost_MSP3=1.2; % Manufacturing cost of MS Pro slot
Power_MSP3=3;  % Power Consumption per hour (Milliamps)

%-------------------------MS DUO Flash Drive-----------------------------

%MANUFACTURER 1: MS DUO Flash Drive
MSD1=SMS_SUB(10); %EXTERNAL STORAGE Discrete Variable
Weight_MSD1=2.4;
Cost_MSD1=1.54; % Manufacturing cost of MS DUO slot
Power_MSD1=2;  % Power Consumption per hour (Milliamps)

%MANUFACTURER 2: MS DUO Flash Drive
MSD2=SMS_SUB(11); %EXTERNAL STORAGE Discrete Variable
Weight_MSD2=2.5;
Cost_MSD2=1.65; % Manufacturing cost of MS DUO slot
Power_MSD2=1.5;  % Power Consumption per hour (Milliamps)

%MANUFACTURER 3: MS DUO Flash Drive
MSD3=SMS_SUB(12);%EXTERNAL STORAGE Discrete Variable
Weight_MSD3=3.0;
Cost_MSD3=1.44; % Manufacturing cost of MS DUO slot
Power_MSD3=2.5;  % Power Consumption per hour (Milliamps)
```

Figure 9.8: Engineering Design Constraints (cont.)


```
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^HARD DRIVE STORAGE^^^^^^^^^^^^^^^^^^^^^^^^^^^

%--------------------------1 Gigabyte HD Storage-----------------------
% MANUFACTURER 1: 1 Gigabyte Storage
GB1HD_1=SMS_SUB(13);
Weight_GB1HD_1=17.2; % 1 GB Component weight
Cost_GB1HD_1=16.80; % Component cost of 1 GB storage device
Power_GB1HD_1=7;  % Power Consumption per hour (Milliamps)


% MANUFACTURER 2: 1 Gigabyte Storage
GB1HD_2=SMS_SUB(14);
Weight_GB1HD_2=19.5; % 1 GB Component weight
Cost_GB1HD_2=15.63; % Component cost of 1 GB storage device
Power_GB1HD_2=7;  % Power Consumption per hour (Milliamps)

% MANUFACTURER 3: 1 Gigabyte Storage
GB1HD_3=SMS_SUB(15);
Weight_GB1HD_3=22.7; % 1 GB Component weight
Cost_GB1HD_3=17.40; % Component cost of 1 GB storage device
Power_GB1HD_3=7;  % Power Consumption per hour (Milliamps)

%--------------------------2 Gigabyte HD Storage-------------------------
% MANUFACTURER 1: 2 Gigabyte Storage
GB2HD_1=SMS_SUB(16);
Weight_GB2HD_1=27.8; % 2 GB Component weight
Cost_GB2HD_1=25.90; % Component cost of 2 GB storage device
Power_GB2HD_1=11;  % Power Consumption per hour (Milliamps)


% MANUFACTURER 2: 2 Gigabyte Storage
GB2HD_2=SMS_SUB(17);
Weight_GB2HD_2=32.9; % 2 GB Component weight
Cost_GB2HD_2=24.83; % Component cost of 2 GB storage device
Power_GB2HD_2=12;  % Power Consumption per hour (Milliamps)

% MANUFACTURER 3: 2 Gigabyte Storage
GB2HD_3=SMS_SUB(18);
Weight_GB2HD_3=28.77; % 2 GB Component weight
Cost_GB2HD_3=26.80; % Component cost of 2 GB storage device
Power_GB2HD_3=13;  % Power Consumption per hour (Milliamps)


  %^^^^^^^^^^^^^^^^^^^^EXTERIOR DESIGN OF CELL PHONE^^^^^^^^^^^^^^^^^^^^

%-----------------------------Shell Phone--------------------------------
SHELL=SMS_SUB(19); % Discrete choice variable: Shell Phone design
Length_SHELL=SMS_SUB(20); % Phone length
Width_SHELL=SMS_SUB(21); % Phone width
Thickness_SHELL=SMS_SUB(22);% Phone Thickness
Weight_SHELL=SMS_SUB(23);% Cell Phone Weight
```

Figure 9.8: Engineering Design Constraints (cont.)


```
Cost_SHELL=SMS_SUB(24);% Cell Phone Cost

%------------------------------Flip Phone---------------------------------

FLIP=SMS_SUB(25); % Discrete choice variable: Flipe Phone design
Length_FLIP=SMS_SUB(26); % Phone length
Width_FLIP=SMS_SUB(27); % Phone width
Thickness_FLIP=SMS_SUB(28);% Phone Thickness
Weight_FLIP=SMS_SUB(29);% Cell Phone Weight
Cost_FLIP=SMS_SUB(30);% Cell Phone Cost

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^BATTERY TYPE^^^^^^^^^^^^^^^^^^^^^^^^^^^^

%      -----------------Nickel Metal Hydride (NIMH)--------------------
NIMH=SMS_SUB(31); %NIMH Discrete Variable
Weight_NIMH=SMS_SUB(32); % Battery Weight
Length_NIMH =SMS_SUB(33); %  Battery Length
Width_NIMH=SMS_SUB(34);   %Battery Width
Thickness_NIMH=SMS_SUB(35);   %Battery Thickness
Cost_NIMH=SMS_SUB(36);  % Battery Cost
Capacity_NIMH=SMS_SUB(37); % Battery Capacity

%   -------------------Lithium Ion (LI-ION)----------------------------

LION=SMS_SUB(38); %LION Discrete Variable
Weight_LION=SMS_SUB(39); % Battery Weight
Length_LION =SMS_SUB(40); %  Battery Length
Width_LION=SMS_SUB(41);   %Battery Width
Thickness_LION=SMS_SUB(42);   %Battery Thickness
Cost_LION=SMS_SUB(43);  % Battery Cost
Capacity_LION=SMS_SUB(44); % Battery Capacity

TALK_TIME=SMS_SUB(45); % Battery Talk Time

%^^^^^^^^^^^^^^^^^^^^^^^^^^CONNECTIVITY^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

%    -------------------------Bluetooth----------------------------


BT=SMS_SUB(46);  % Bluetooth Discrete Variable
Weight_BT=8.9; % Weight of Bluetooth radio
Cost_BT=5.80;  % Cost of Bluetooth radio
Power_BT=8;   % Power consumption rate of radio


%    ^^^^^^^^^^^^^^^^WiFi^^^^^^^^^^^^^^^^^^^

WF=SMS_SUB(47);  % Wifi Discrete Variable
Weight_WF=12.8; % Weight of WiFi radio
Cost_WF=7.30;  % Cost of Wifi radio
Power_WF=8;   % Power consumption rate Wifi
```

Figure 9.8: Engineering Design Constraints (cont.)


```
%    ^^^^^^^^^^^^^^^^Infra-red^^^^^^^^^^^^^^^^^^^

IR=SMS_SUB(48);  % Infra-red Discrete Variable
Weight_IR=8.5; % Weight of Infra red
Cost_IR=3.73;  % Cost of Infra red
Power_IR=6;  % Power consumption rate of Infra-red


%----------------------------Audio Codec-------------------------------%

%    ^^^^^^^^^^^^^^^^^^^^^^^^^Microphone^^^^^^^^^^^^^^^^^^^^^^^^^

% MANUFACTURER 1: Microphone
MIC1=SMS_SUB(49); %Microphone Discrete Variable
Weight_MIC1=2.1; % Weight of Microphone
Cost_MIC1=0.81;  % Cost of Microphone
Power_MIC1=.4;   % Power consumption rate of Microphone

% MANUFACTURER 2: Microphone
MIC2=SMS_SUB(50); %Microphone Discrete Variable
Weight_MIC2=2.6; % Weight of Microphone
Cost_MIC2=0.84;  % Cost of Microphone
Power_MIC2=.7;   % Power consumption rate of Microphone

%    ^^^^^^^^^^^^^^^^^^^^^^^^^Earpiece^^^^^^^^^^^^^^^^^^^^^^^^^

% MANUFACTURER 1: Earpiece
EP1=SMS_SUB(51); %Microphone Discrete Variable
Weight_EP1=2.7; % Weight of Earpiece
Cost_EP1=0.13;  % Cost of Earpiece
Power_EP1=1.1;   % Power consumption rate of Earpiece

% MANUFACTURER 2: Earpiece
EP2=SMS_SUB(52); %Microphone Discrete Variable
Weight_EP2=2.4; % Weight of Earpiece
Cost_EP2=0.14;  % Cost of Earpiece
Power_EP2=1.3;   % Power consumption rate of Earpiece

% MANUFACTURER 3: Earpiece
EP3=SMS_SUB(53); %Microphone Discrete Variable
Weight_EP3=3.1; % Weight of Earpiece
Cost_EP3=0.14;  % Cost of Earpiece
Power_EP3=1.0;   % Power consumption rate of Earpiece


%    ^^^^^^^^^^^^^^^^^^^^^^^^^Audio Jack^^^^^^^^^^^^^^^^^^^^^^^^^

% MANUFACTURER 1: Audio Jack
AJ1=SMS_SUB(54);  % Audio Jack Discrete Variable
Weight_AJ1=1.4; % Weight of Audio Jack
Cost_AJ1=0.8;  % Cost of Audio Jack
Power_AJ1=0.5;
```

Figure 9.8: Engineering Design Constraints (cont.)


```
% MANUFACTURER 2: Audio Jack
AJ2=SMS_SUB(55);  % Audio Jack Discrete Variable
Weight_AJ2=1.7; % Weight of Audio Jack
Cost_AJ2=0.6;  % Cost of Audio Jack
Power_AJ2=0.9;
%    ^^^^^^^^^^^^^^^^^^^^^^^^^External Speaker^^^^^^^^^^^^^^^^^^^^^^^^^

% MANUFACTURER 1: External Speaker
ES1=SMS_SUB(56); %External Speaker Discrete Variable
Weight_ES1=8.43; % Weight of External Speaker
Cost_ES1=2.9;  % Cost of External Speaker
Power_ES1=1.8;   % Power consumption rate of External Speaker

% MANUFACTURER 2: External Speaker
ES2=SMS_SUB(57); %External Speaker Discrete Variable
Weight_ES2=7.12; % Weight of External Speaker
Cost_ES2=3.7;  % Cost of External Speaker
Power_ES2=1.3;   % Power consumption rate of External Speaker

% MANUFACTURER 3: External Speaker
ES3=SMS_SUB(58); %External Speaker Discrete Variable
Weight_ES3=6.9; % Weight of External Speaker
Cost_ES3=1.7;  % Cost of External Speaker
Power_ES3=1.5;   % Power consumption rate of External Speaker

%---------------------------------LCD---------------------------------%

%    ^^^^^^^^^^^^^^^^^^^^^^^^^TFT LCD^^^^^^^^^^^^^^^^^^^^^^^^^

LCD=SMS_SUB(59); % TFT LCD Discrete Choice variable
LCD_length= SMS_SUB(60); % Length of LCD
LCD_width=  SMS_SUB(61); % Width of LCD
LCD_res=    SMS_SUB(62); % Display Resolution
Cost_LCD=      SMS_SUB(63); % LCD manufacturing cost
Weight_LCD=      SMS_SUB(64); % LCD Unit weight
Power_LCD=      SMS_SUB(65); % Battery Consumption;


%    ^^^^^^^^^^^^^^^^^^^^^^^^^OLED^^^^^^^^^^^^^^^^^^^^^^^^^

OLED=SMS_SUB(66); % OLED Discrete Choice variable
OLED_length= SMS_SUB(67); % Length of OLED
OLED_width=  SMS_SUB(68); % Width of OLED
OLED_res=    SMS_SUB(69); % Display Resolution
Cost_OLED=      SMS_SUB(70); % OLED manufacturing cost
Weight_OLED=      SMS_SUB(71); % OLED Unit weight
Power_OLED=      SMS_SUB(72); % Battery Consumption;


%*************************Digital Camera Modules************************
```

Figure 9.8: Engineering Design Constraints (cont.)


```
%    ^^^^^^^^^^^^^^^^^^^^^^^^^1 MegaPixel Camera^^^^^^^^^^^^^^^^^^^^^^^^

%MANUFACTURER 1: 1 MP Camera Module
One_MP1=SMS_SUB(73); %1 MP Discrete Choice variable
Cost_One_MP1=9.75; %1 MP Camera Component Cost
Weight_One_MP1=15.2;  % 1 MP Camera Component Weight
Power_One_MP1=5.7;  % 1 MP Camera Component Power consumption

%MANUFACTURER 2: 1 MP Camera Module
One_MP2=SMS_SUB(74); %1 MP Discrete Choice variable
Cost_One_MP2=7.75; %1 MP Camera Component Cost
Weight_One_MP2=15.15;  % 1 MP Camera Component Weight
Power_One_MP2=5.4;  % 1 MP Camera Component Power consumption

%    ^^^^^^^^^^^^^^^^^^^^^^^^^2 MegaPixel Camera^^^^^^^^^^^^^^^^^^^^^^^^

%MANUFACTURER 1: 2 MP Camera Module
Two_MP1=SMS_SUB(75); %2 MP Discrete Choice variable
Cost_Two_MP1=11.75; %2 MP Camera Component Cost
Weight_Two_MP1=14.7;  % 2 MP Camera Component Weight
Power_Two_MP1=5.9;  % 2 MP Camera Component Power consumption

%MANUFACTURER 2: 2 MP Camera Module
Two_MP2=SMS_SUB(76); %2 MP Discrete Choice variable
Cost_Two_MP2=12.33; %2 MP Camera Component Cost
Weight_Two_MP2=15.1;  % 2 MP Camera Component Weight
Power_Two_MP2=6.1;  % 2 MP Camera Component Power consumption


%*************************MP3 Modules************************

%MANUFACTURER 1:  MP3 Module
MP3_1=SMS_SUB(77); % MP3 Discrete Choice variable
Cost_MP3_1=9.75; %  MP3 Component Cost
Weight_MP3_1=8.43;  %MP3 Component Weight
Power_MP3_1=5.7;  % MP3 Component Power consumption

%MANUFACTURER 2:  MP3 Module
MP3_2=SMS_SUB(78); % MP3 Discrete Choice variable
Cost_MP3_2=8.22; %  MP3 Component Cost
Weight_MP3_2=7.22;  %MP3 Component Weight
Power_MP3_2=5.9;  % MP3 Component Power consumption

%MANUFACTURER 3:  MP3 Module
MP3_3=SMS_SUB(79); % MP3 Discrete Choice variable
Cost_MP3_3=9.6; %  MP3 Component Cost
Weight_MP3_3=9.93;  %MP3 Component Weight
Power_MP3_3=5.5;  % MP3 Component Power consumption

%MANUFACTURER 4:  MP3 Module
MP3_4=SMS_SUB(80); % MP3 Discrete Choice variable
Cost_MP3_4=9.2; %  MP3 Component Cost
```

Figure 9.8: Engineering Design Constraints (cont.)


```
Weight_MP3_4=1.3;  %MP3 Component Weight
Power_MP3_4=4.3;  % MP3 Component Power consumption

%**********************Module for Internet Capabilities*****************

%MANUFACTURER 1:  Internet Module
INT_1=SMS_SUB(81); % INT Discrete Choice variable
Cost_INT_1=4.2; %  INT Component Cost
Weight_INT_1=10.8;  %INT Component Weight
Power_INT_1=4.7;  % INT Component Power consumption

%MANUFACTURER 2:  Internet Module
INT_2=SMS_SUB(82); % INT Discrete Choice variable
Cost_INT_2=3.72; %  INT Component Cost
Weight_INT_2=11.4;  %INT Component Weight
Power_INT_2=5.1;  % INT Component Power consumption

%******************Graphics Module for Games**************************
%MANUFACTURER 1:      GAMES Module
GAMES1=SMS_SUB(83); %   GAMES Discrete Choice variable
Cost_GAMES1=6.25; %   GAMES Component Cost
Weight_GAMES1=11.8;  % GAMES Component Weight
Power_GAMES1=9.3;  %  GAMES Component Power consumption

%MANUFACTURER 2:      GAMES Module
GAMES2=SMS_SUB(84); %   GAMES Discrete Choice variable
Cost_GAMES2=6.1; %   GAMES Component Cost
Weight_GAMES2=9.6;  % GAMES Component Weight
Power_GAMES2=8.2;  %  GAMES Component Power consumption

%MANUFACTURER 3:      GAMES Module
GAMES3=SMS_SUB(85); %   GAMES Discrete Choice variable
Cost_GAMES3=5.95; %   GAMES Component Cost
Weight_GAMES3=11.5;  % GAMES Component Weight
Power_GAMES3=8.5;  %  GAMES Component Power consumption

%***********************SMS TEXT CAPABLE PHONE**************************
%MANUFACTURER 1:      SMS Module
SMS=SMS_SUB(86); %   SMS Discrete Choice variable
Cost_SMS=4.75; %   SMS Component Cost
Weight_SMS=12.5;  % SMS Component Weight
Power_SMS=2.1;  %  SMS Component Power consumption
SMS_Key_Tol=40;

%**************Overall Architecture Design Variables*******************
 Total_Cost=SMS_SUB(87);       %Total Product Architecture Cost
 Total_Weight=SMS_SUB(88);

%---------------------------CONSTANTS--------------------------------%
  B1=18.21;    %Assumed cost of components not included in the model

  A1=14.74;  %Unit pixel/Area based on standard phone specs TFT LCD
```

Figure 9.8: Engineering Design Constraints (cont.)


```
  A2=5*10^-3; %TFT LCD manufacturing cost/unit volume
  A3=0.04;     %TFT LCD Unit weight
  A4=0.01;   %TFT battery consumption per unit resolution

  A5=19.62;  %Unit pixel/Area based on standard phone specs OLED LCD
  A6=8*10^-3; %OLED LCD manufacturing cost/unit volume
  A7=0.03;     %OLED LCD Unit weight
  A8=0.03;   %OLED battery consumption per unit resolution


  NIMH_Const1=0.021374687; %BATTERY CONSTANTS
  NIMH_Const2=0.000379288; %NIMH Cost Constant
  NIMH_Const3=9.38e-4;     %NIMH Weight Constant

  LION_Const1=0.043186444; %BATTERY CONSTANTS
  LION_Const2=0.000803447; %LION Cost Constant
  LION_Const3=8.83e-4;     %LION Weight Constant

  SHELL_Const1=2.29e-4;  %Shell phone unit Cost/volume
  SHELL_Const2=5.1e-4;  %Shell phone unit Weight/volume

  FLIP_Const1=1.47e-4;%Flip  phone unit Cost/volume
  FLIP_Const2=4.9e-4;%Flip  phone unit Weight/volume



  %^^^^^^^^^^^^^^^^^^^^^^^^^^CONSTRAINTS^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  %**************************TFT LCD************************************
  h1=(A1*LCD_length*LCD_width)-LCD_res;    %LCD resolution
  h2=(A2*LCD_length*LCD_width)-Cost_LCD ;  %LCD Manufacturing Cost
  h3=(A3*LCD_length*LCD_width)-Weight_LCD;  %LCD component Weight
  g1=LCD_length-(0.60*SHELL*Length_SHELL+0.60*FLIP*Length_FLIP);        %Length cannot↙
exceed a certain % of Phone length
  g2=(0.30*SHELL*Length_SHELL+0.30*FLIP*Length_FLIP)-LCD_length;        %Length is >↙
certain % of Phone length
  g3=LCD_width-0.90*(SHELL*Width_SHELL+FLIP*Width_FLIP);                %Width cannot↙
exceed a certain % of Phone length
  g4=0.7*(SHELL*Width_SHELL+FLIP*Width_FLIP)-LCD_width;                 %Width is >↙
certain % of Phone length
  h4=(A4*LCD_length*LCD_width)-Power_LCD;                               %TFT LCD battery↙
consumption based on size

 %**************************OLED LCD************************************
  h5=(A5*OLED_length*OLED_width)-OLED_res;    %OLED resolution
  h6=(A6*OLED_length*OLED_width)-Cost_OLED ;  %OLED Manufacturing Cost
  h7=(A7*OLED_length*OLED_width)-Weight_OLED;  %OLED component Weight
  g5=OLED_length-(0.60*SHELL*Length_SHELL+0.60*FLIP*Length_FLIP);        %Length cannot↙
exceed a certain % of Phone length
  g6=(0.30*SHELL*Length_SHELL+0.30*FLIP*Length_FLIP)-OLED_length;        %Length is >↙
certain % of Phone length
  g7=OLED_width-0.90*(SHELL*Width_SHELL+FLIP*Width_FLIP);                %Width cannot↙
exceed a certain % of Phone length
```

Figure 9.8: Engineering Design Constraints (cont.)


```
  g8=0.7*(SHELL*Width_SHELL+FLIP*Width_FLIP)-OLED_width;                %Width is >↙
certain % of Phone length
  h8=(A8*OLED_length*OLED_width)-Power_OLED;  %OLED battery consumption based on size


%**********************NIMH BATTERY  Capacity Caculation*********************
  h9=Capacity_NIMH-((NIMH_Const1*(Length_NIMH*Width_NIMH*Thickness_NIMH)));

  g9= (TALK_TIME*(Mbyte_32_1*Power_32MB1+Mbyte_32_2*Power_32MB2+Mbyte_32_3*Power_32MB3...
   +Mbyte_64_1*Power_64MB1+Mbyte_64_2*Power_64MB2+Mbyte_64_3*Power_64MB3...
   +MSP1*Power_MSP1+MSP2*Power_MSP2+MSP3*Power_MSP3+MSD1*Power_MSD1+MSD2*Power_MSD2...
   +MSD3*Power_MSD3+GB1HD_1*Power_GB1HD_1+GB1HD_2*Power_GB1HD_2+GB1HD_3*Power_GB1HD_3...
↙
+GB2HD_1*Power_GB2HD_1+GB2HD_2*Power_GB2HD_2+GB2HD_3*Power_GB2HD_3+BT*Power_BT+WF*Power_WF↙
+IR*Power_IR...
   +MIC1*Power_MIC1+MIC2*Power_MIC2+EP1*Power_EP1+EP2*Power_EP2+EP3*Power_EP3...
   +AJ1*Power_AJ1+AJ2*Power_AJ2+ES1*Power_ES1+ES2*Power_ES2+ES3*Power_ES3...
   +LCD*Power_LCD+OLED*Power_OLED+One_MP1*Power_One_MP1+One_MP2*Power_One_MP2+↙
Two_MP1*Power_Two_MP1...
   +Two_MP2*Power_Two_MP2+ MP3_1*Power_MP3_1+MP3_2*Power_MP3_2+MP3_3*Power_MP3_3...
   +MP3_4*Power_MP3_4+INT_1*Power_INT_1+INT_2*Power_INT_2+GAMES1*Power_GAMES1...
   +GAMES2*Power_GAMES2+GAMES3*Power_GAMES3+SMS*Power_SMS))-Capacity_NIMH;

%**********************LION BATTERY  Capacity Caculation*********************
  h10=Capacity_LION-((LION_Const1*(Length_LION*Width_LION*Thickness_LION)));

  g10=(TALK_TIME*(Mbyte_32_1*Power_32MB1+Mbyte_32_2*Power_32MB2+Mbyte_32_3*Power_32MB3...
   +Mbyte_64_1*Power_64MB1+Mbyte_64_2*Power_64MB2+Mbyte_64_3*Power_64MB3...
   +MSP1*Power_MSP1+MSP2*Power_MSP2+MSP3*Power_MSP3+MSD1*Power_MSD1+MSD2*Power_MSD2...
   +MSD3*Power_MSD3+GB1HD_1*Power_GB1HD_1+GB1HD_2*Power_GB1HD_2+GB1HD_3*Power_GB1HD_3...
↙
+GB2HD_1*Power_GB2HD_1+GB2HD_2*Power_GB2HD_2+GB2HD_3*Power_GB2HD_3+BT*Power_BT+WF*Power_WF↙
+IR*Power_IR...
   +MIC1*Power_MIC1+MIC2*Power_MIC2+EP1*Power_EP1+EP2*Power_EP2+EP3*Power_EP3...
   +AJ1*Power_AJ1+AJ2*Power_AJ2+ES1*Power_ES1+ES2*Power_ES2+ES3*Power_ES3...
   +LCD*Power_LCD+OLED*Power_OLED+One_MP1*Power_One_MP1+One_MP2*Power_One_MP2+↙
Two_MP1*Power_Two_MP1...
   +Two_MP2*Power_Two_MP2+ MP3_1*Power_MP3_1+MP3_2*Power_MP3_2+MP3_3*Power_MP3_3...
   +MP3_4*Power_MP3_4+INT_1*Power_INT_1+INT_2*Power_INT_2+GAMES1*Power_GAMES1...
   +GAMES2*Power_GAMES2+GAMES3*Power_GAMES3+SMS*Power_SMS))-Capacity_LION;

  g11=(NIMH*Length_NIMH+LION*Length_LION)-0.60*(SHELL*Length_SHELL+FLIP*Length_SHELL); %↙
Battery Length must be less than a certain % of cell phone
  g12=(NIMH*Width_NIMH+LION*Width_LION)-0.95*(SHELL*Width_SHELL+FLIP*Width_FLIP);%Battery↙
Width must be less than a certain % of cell phone
  g13=(NIMH*Thickness_NIMH+LION*Thickness_LION)-0.45*↙
(SHELL*Thickness_SHELL+FLIP*Thickness_FLIP);% Battery thickness must be less than %

  %**********************BATTERY  Talk Time Caculation******************
  h11=TALK_TIME-(NIMH*((0.0053*(Capacity_NIMH))+0.0269)+ (LION*((0.0061*(Capacity_LION))↙
+0.1667)));
```

Figure 9.8: Engineering Design Constraints (cont.)


```
  %**********************BATTERY  Cost Caculation******************
  h12=((NIMH_Const2*(Length_NIMH*Width_NIMH*Thickness_NIMH))-Cost_NIMH);
  h13=((LION_Const2*(Length_LION*Width_LION*Thickness_LION))-Cost_LION);


  %**********************BATTERY  Weight Caculation******************
  h14=((NIMH_Const3*(Length_NIMH*Width_NIMH*Thickness_NIMH))-Weight_NIMH);
  h15=((LION_Const3*(Length_LION*Width_LION*Thickness_LION))-Weight_LION);

  %_____________________CELL PHONE DESIGN CONSTRAINTS_____________________
  h16=(SHELL_Const1*Length_SHELL*Width_SHELL*Thickness_SHELL)-Cost_SHELL;
  h17=(FLIP_Const1*Length_FLIP*Width_FLIP*Thickness_FLIP)-Cost_FLIP;
  h18=(SHELL_Const2*Length_SHELL*Width_SHELL*Thickness_SHELL)-Weight_SHELL;
  h19=(FLIP_Const2*Length_FLIP*Width_FLIP*Thickness_FLIP)-Weight_FLIP;


  %****************Constraint for cost of Product Platform*****************
  h20=Total_Cost-(B1+Mbyte_32_1*Cost_32MB1+Mbyte_32_2*Cost_32MB2+Mbyte_32_3*Cost_32MB3...
   +Mbyte_64_1*Cost_64MB1+Mbyte_64_2*Cost_64MB2+Mbyte_64_3*Cost_64MB3...
   +MSP1*Cost_MSP1+MSP2*Cost_MSP2+MSP3*Cost_MSP3+MSD1*Cost_MSD1+MSD2*Cost_MSD2...
   +MSD3*Cost_MSD3+GB1HD_1*Cost_GB1HD_1+GB1HD_2*Cost_GB1HD_2+GB1HD_3*Cost_GB1HD_3...
   +GB2HD_1*Cost_GB2HD_1+GB2HD_2*Cost_GB2HD_2+GB2HD_3*Cost_GB2HD_3+SHELL*Cost_SHELL...
   +FLIP*Cost_FLIP+NIMH*Cost_NIMH+LION*Cost_LION+BT*Cost_BT+WF*Cost_WF+IR*Cost_IR...
   +MIC1*Cost_MIC1+MIC2*Cost_MIC2+EP1*Cost_EP1+EP2*Cost_EP2+EP3*Cost_EP3...
   +AJ1*Cost_AJ1+AJ2*Cost_AJ2+ES1*Cost_ES1+ES2*Cost_ES2+ES3*Cost_ES3...
   +LCD*Cost_LCD+OLED*Cost_OLED+One_MP1*Cost_One_MP1+One_MP2*Cost_One_MP2+↙
Two_MP1*Cost_Two_MP1...
   +Two_MP2*Cost_Two_MP2+    MP3_1*Cost_MP3_1+MP3_2*Cost_MP3_2+MP3_3*Cost_MP3_3...
   +MP3_4*Cost_MP3_4+INT_1*Cost_INT_1+INT_2*Cost_INT_2+GAMES1*Cost_GAMES1...
   +GAMES2*Cost_GAMES2+GAMES3*Cost_GAMES3+SMS*Cost_SMS);

  h21= Total_Weight-↙
(Mbyte_32_1*Weight_32MB1+Mbyte_32_2*Weight_32MB2+Mbyte_32_3*Weight_32MB3...
   +Mbyte_64_1*Weight_64MB1+Mbyte_64_2*Weight_64MB2+Mbyte_64_3*Weight_64MB3...
↙
+MSP1*Weight_MSP1+MSP2*Weight_MSP2+MSP3*Weight_MSP3+MSD1*Weight_MSD1+MSD2*Weight_MSD2...
↙
+MSD3*Weight_MSD3+GB1HD_1*Weight_GB1HD_1+GB1HD_2*Weight_GB1HD_2+GB1HD_3*Weight_GB1HD_3...
↙
+GB2HD_1*Weight_GB2HD_1+GB2HD_2*Weight_GB2HD_2+GB2HD_3*Weight_GB2HD_3+SHELL*Weight_SHELL..↙
.
↙
+FLIP*Weight_FLIP+NIMH*Weight_NIMH+LION*Weight_LION+BT*Weight_BT+WF*Weight_WF+IR*Weight_IR↙
...
   +MIC1*Weight_MIC1+MIC2*Weight_MIC2+EP1*Weight_EP1+EP2*Weight_EP2+EP3*Weight_EP3...
   +AJ1*Weight_AJ1+AJ2*Weight_AJ2+ES1*Weight_ES1+ES2*Weight_ES2+ES3*Weight_ES3...
   +LCD*Weight_LCD+OLED*Weight_OLED+One_MP1*Weight_One_MP1+One_MP2*Weight_One_MP2+↙
Two_MP1*Weight_Two_MP1...
   +Two_MP2*Weight_Two_MP2+MP3_1*Weight_MP3_1+MP3_2*Weight_MP3_2+MP3_3*Weight_MP3_3...
   +MP3_4*Weight_MP3_4+INT_1*Weight_INT_1+INT_2*Weight_INT_2+GAMES1*Weight_GAMES1...
   +GAMES2*Weight_GAMES2+GAMES3*Weight_GAMES3+SMS*Weight_SMS);
```

Figure 9.8: Engineering Design Constraints (cont.)


```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SMS_SUB(1)=Mbyte_32_1;
SMS_SUB(2)=Mbyte_32_2;
SMS_SUB(3)=Mbyte_32_3;
SMS_SUB(4)=Mbyte_64_1;
SMS_SUB(5)=Mbyte_64_2;
SMS_SUB(6)=Mbyte_64_3;

SMS_SUB(7)=MSP1;
SMS_SUB(8)=MSP2;
SMS_SUB(9)=MSP3;
SMS_SUB(10)=MSD1;
SMS_SUB(11)=MSD2;
SMS_SUB(12)=MSD3;

SMS_SUB(13)=GB1HD_1;
SMS_SUB(14)=GB1HD_2;
SMS_SUB(15)=GB1HD_3;
SMS_SUB(16)=GB2HD_1;
SMS_SUB(17)=GB2HD_2;
SMS_SUB(18)=GB2HD_3;

SMS_SUB(19)=SHELL; % Discrete choice variable: Shell Phone design
SMS_SUB(20)=Length_SHELL; % Phone length
SMS_SUB(21)=Width_SHELL; % Phone width
SMS_SUB(22)=Thickness_SHELL;% Phone Thickness
SMS_SUB(23)=Weight_SHELL;% Cell Phone Weight
SMS_SUB(24)=Cost_SHELL;% Cell Phone Cost

SMS_SUB(25)=FLIP; % Discrete choice variable: Flipe Phone design
SMS_SUB(26)=Length_FLIP; % Phone length
SMS_SUB(27)=Width_FLIP; % Phone width
SMS_SUB(28)=Thickness_FLIP;% Phone Thickness
SMS_SUB(29)=Weight_FLIP;% Cell Phone Weight
SMS_SUB(30)=Cost_FLIP;% Cell Phone Cost

%      -----------------Nickel Metal Hydride (NIMH)---------------------
SMS_SUB(31)=NIMH; %NIMH Discrete Variable
SMS_SUB(32)=Weight_NIMH; % Battery Weight
SMS_SUB(33)=Length_NIMH ; %  Battery Length
SMS_SUB(34)=Width_NIMH;   %Battery Width
SMS_SUB(35)=Thickness_NIMH;   %Battery Thickness
SMS_SUB(36)=Cost_NIMH;  % Battery Cost
SMS_SUB(37)=Capacity_NIMH; % Battery Capacity

%   --------------------Lithium Ion (LI-ION)-----------------------------

SMS_SUB(38)=LION; %LION Discrete Variable
SMS_SUB(39)=Weight_LION; % Battery Weight
SMS_SUB(40)=Length_LION ; %  Battery Length
SMS_SUB(41)=Width_LION;   %Battery Width
SMS_SUB(42)=Thickness_LION;   %Battery Thickness
```

Figure 9.8: Engineering Design Constraints (cont.)


```
SMS_SUB(43)=Cost_LION;  % Battery Cost
SMS_SUB(44)=Capacity_LION; % Battery Capacity
SMS_SUB(45)=TALK_TIME; % Battery Talk Time

SMS_SUB(46)=BT;
SMS_SUB(47)=WF;
SMS_SUB(48)=IR;

SMS_SUB(49)=MIC1;
SMS_SUB(50)=MIC2;

SMS_SUB(51)=EP1;
SMS_SUB(52)=EP2;
SMS_SUB(53)=EP3;

SMS_SUB(54)=AJ1;
SMS_SUB(55)=AJ2;

SMS_SUB(56)=ES1;
SMS_SUB(57)=ES2;
SMS_SUB(58)=ES3;

SMS_SUB(59)=LCD; % TFT LCD Discrete Choice variable
SMS_SUB(60)=LCD_length; % Length of LCD
SMS_SUB(61)=LCD_width; % Width of LCD
SMS_SUB(62)=LCD_res; % Display Resolution
SMS_SUB(63)=Cost_LCD; % LCD manufacturing cost
SMS_SUB(64)=Weight_LCD; % LCD Unit weight
SMS_SUB(65)=Power_LCD; % Battery Consumption;


%    ^^^^^^^^^^^^^^^^^^^^^^^^^OLED^^^^^^^^^^^^^^^^^^^^^^^^^

SMS_SUB(66)=OLED; % OLED Discrete Choice variable
SMS_SUB(67)=OLED_length; % Length of OLED
SMS_SUB(68)=OLED_width; % Width of OLED
SMS_SUB(69)=OLED_res; % Display Resolution
SMS_SUB(70)=Cost_OLED; % OLED manufacturing cost
SMS_SUB(71)=Weight_OLED; % OLED Unit weight
SMS_SUB(72)=Power_OLED; % Battery Consumption;

SMS_SUB(73)=One_MP1;
SMS_SUB(74)=One_MP2;

SMS_SUB(75)=Two_MP1;
SMS_SUB(76)=Two_MP2;

SMS_SUB(77)=MP3_1;
SMS_SUB(78)=MP3_2;
SMS_SUB(79)=MP3_3;
SMS_SUB(80)=MP3_4;
```

Figure 9.9: Engineering Variable bound definition

```
7/12/07 1:10 PM  C:\MATLAB\work\ASME_matlab\ASME_DAC1\Optimal P...\SMS_SUBSYS2_1.m   1 of 3

load X_ALL2_1.mat X_ALL2_1 -ascii ;

global target_SUB,count;
%Starting Point
x0(1:88)=X_ALL2_1(1:88);


%Variable Lower Bounds
vl(1)=0;  vl(11)=0;  vl(21)=40;  vl(31)=0;   vl(40)=0;    vl(50)=0;     vl(60)=0;↙
vl(70)=0;           vl(80)=0;
vl(2)=0;  vl(12)=0;   vl(22)=12;  vl(32)=0;   vl(41)=0;    vl(51)=0;    vl(61)=0;↙
vl(71)= 0;          vl(81)=0;
vl(3)=0;  vl(13)=0;   vl(23)=0;   vl(33)=0;   vl(42)=0;     vl(52)=0;    vl(62)=5000;↙
vl(72)= 0;          vl(82)=0;
vl(4)=0;   vl(14)=0;  vl(24)=0;   vl(34)=0;   vl(43)=0;     vl(53)=0;    vl(63)=0;↙
vl(73)=0;           vl(83)=0;
vl(5)=0;  vl(15)=0;  vl(25)=0;   vl(35)=0;    vl(44)=200;     vl(54)=0;    vl(64)=0;↙
vl(74)=0;           vl(84)=0;
vl(6)=0;  vl(16)=0;  vl(26)=100;   vl(36)=0;   vl(45)=0;     vl(55)=0;      vl(65)=0;↙
vl(75)=0;           vl(85)=0;
vl(7)=0;  vl(17)=0;   vl(27)=45;  vl(37)=300;   vl(46)=0;    vl(56)=0;      vl(66)=0;↙
vl(76)=0;           vl(86)=0;
vl(8)=0; vl(18)=0;   vl(28)=12;   vl(38)=0;     vl(47)=0;    vl(57)=0;       vl(67) =0;↙
vl(77)=0;           vl(87)=0;
vl(9)=0; vl(19)=0;   vl(29)=0;     vl(39)=0;     vl(48)=0;     vl(58)=0;   vl(68) =0;↙
vl(78)=0;            vl(88)=0;
vl(10)=0; vl(20)=80;  vl(30)=0;                 vl(49)=0;    vl(59)=0;    vl(69) =5000;↙
vl(79)=0;


%Variable Upper Bounds
vu(1)=1;  vu(11)=1;  vu(21)=70;  vu(31)=1;   vu(40)=80;    vu(50)=1;     vu(60)=70;↙
vu(70)=60;           vu(80)=1;
vu(2)=1;  vu(12)=1;   vu(22)=25;  vu(32)=50;   vu(41)=60;    vu(51)=1;    vu(61)=50;↙
vu(71)= 40;          vu(81)=1;
vu(3)=1;  vu(13)=1;   vu(23)=40;   vu(33)=80;   vu(42)=30;     vu(52)=1;    vu(62)=30000;↙
vu(72)= 30;          vu(82)=1;
vu(4)=1;   vu(14)=1;  vu(24)=35;   vu(34)=60;   vu(43)=100;     vu(53)=1;    vu(63)=60;↙
vu(73)=1;           vu(83)=1;
vu(5)=1;  vu(15)=1;  vu(25)=1;   vu(35)=30;   vu(44)=3000;     vu(54)=1;    vu(64)=40;↙
vu(74)=1;           vu(84)=1;
vu(6)=1;  vu(16)=1;  vu(26)=160;   vu(36)=100;   vu(45)=15;     vu(55)=1;      vu(65)=30;↙
vu(75)=1;         vu(85)=1;
vu(7)=1;  vu(17)=1;   vu(27)=70;  vu(37)=3000;   vu(46)=1;    vu(56)=1;      vu(66)=1;↙
vu(76)=1;            vu(86)=1;
vu(8)=1; vu(18)=1;   vu(28)=25;     vu(38)=1;     vu(47)=1;    vu(57)=1;       vu(67) =70;↙
vu(77)=1;          vu(87)=400;
vu(9)=1; vu(19)=1;   vu(29)=40;      vu(39)=50;    vu(48)=1;     vu(58)=1;   vu(68) =50;↙
vu(78)=1;          vu(88)=1000;
vu(10)=1; vu(20)=120;  vu(30)=50;                 vu(49)=1;    vu(59)=1;    vu(69)↙
=30000;     vu(79)=1;
```