Optimization based framework for transforming automotive configurations for production planning

Tilak Raj Singh [1] and **Narayan Rangaraj** [2]

Abstract. A product (e.g. automobile, computer) can be configured using different combinations of its available attributes (features). However, selection of attributes may not be independent of the selection of other attributes. In practice, each attribute implies a selection rule (dependency) for other sets of attributes in order to generate a valid configuration. Due to dynamic changes in the product design, miniaturization, legislation etc., product attributes and their selection rules get changed. This implies that variants produced in the past may not be valid for future product design. Nevertheless, customer history contains important information related to customer buying behaviour which is an essential input for future planning activities. In order to achieve efficient adaption of past customer orders to a changed product design, we propose a fully automated optimization based framework. The methodology is demonstrated using an industry size example.

1 Introduction

Manufacturing companies are currently focusing on mass customization. In this environment customer mix and match different available product attributes to get desired configurations. Selection of any attribute implies certain conditions on other set of attributes. For example, if driving assistance system is selected in a car then the customer may only be able to select steering types which have required control options. These engineering dependencies are available in the product's technical documentation (e.g. Bill-Of-Material) and each valid configuration must satisfy these restrictions in order to be producible [13]. Manufacturers enable their customers to select and order constructible product variants by offering sales manuals and web-based product configurators. The product configurator guarantees that attributes selected by the customer must satisfy all dependency rules at the time configuration is created. If any combination of attributes violates the product configuration rules (constraints), then, this will be an invalid configuration and cannot be produced [5]. We will use the term configurations rule (or rules in short) in the meaning of all restrictions imposed on configuration problem and by fulfilling all rules configuration will be considered feasible for production.

In order to provide short lead time for complex engineering products (e.g. Automobiles, Computers) often hybrid manufacturing philosophies like assemble-to-order is used. The production is setup based on forecast demand and final assembly is done for the real customer orders. The effectiveness of this method depends upon the quality of the forecasted demand. Most manufacturers use data about product variants produced in the past to get suitable estimates for the future customer demand [10]. Continuous changes in product design and market conditions imply that product variants which have been produced in the past may not be valid according to the changed product. However, changes are incremental in nature which means that past variants can be upgraded (by dropping and/or adding some features) to new changed model once the required changes are incorporated [4]. In this paper our aim is to develop methods to upgrade base configuration (configuration produced in the past) in such a way that 1) new configuration satisfies required product configuration rules 2) new configuration should be as similar as possible to the base configuration. The similarity measure can be monitored by using some distance (e.g. Hamming distance) or cost function.

In contrast to the above problem, another requirement to transform existing configuration to the new configuration arises from the *Reconfiguration* problem [11]. In this case the previously selected base configuration is still valid with respect to configuration model however the customer may want to make some explicit changes with respect to the earlier choice - for example, adding or dropping some of product's features. Most of these reconfiguration problems are motivated by the customer's request to change the previously selected variants. This is not an uncommon situation in premium customizable products. However, the reconfiguration problem can also be driven from the manufacturer point of view. For example due to capacity limitations, production of customer orders may be shifted from one country/plant to another. Then the production feasibility need to be checked as configuration rules may vary between production plants and counties.

In this paper we propose an integrated solution framework where: 1) user can update any given configuration by changing configuration variables (adding or removing product attributes) 2) Feasibility

[1] IT-Production Tools, Mercedes-Benz R & D India, Bangalore, Email: tilak.singh@daimler.com
[2] Indian Institute of Technology, Bombay, Powai Mumbai, India, email: narayan.rangaraj@iitb.ac.in

of desired configuration can be checked at any point of time 3) In case of conflicts with underlying configuration rules, the solution is computed through solving an optimization model which ensures that the modification to the base configuration is done with minimal change cost. In section 2 and section 3 we will discuss characteristics of the problem and the available data. Section 4 will focus on the development of an optimization based configuration transformation model. In section 5 the solution procedure will be discussed with initial computational results.

2 The planning problem

A product can be configured using different combinations of its attributes (features). In case of automobiles, attributes could be: body style, transmission type, sunroof, parking assistance etc. If we describe a product as an exhaustive list of attributes then the product configuration can be expressed as a 0-1 vector over the attribute set, where 0 (zero) represents the absence of any attribute and 1 (one) represents its presence in the configuration. A feasible configuration can be achieve by satisfying predefined set of rules (Boolean formulas) monitoring interdependencies among attributes.

Let us define our product configuration problem as per [7, Definition 1]: the configuration problem C can be expressed through a triple (X, D, F), where:

- X is a set of product attributes (configuration variables) lets say $\{1, ...n\}$. Where n is the total number of attributes.
- D is the set of attributes finite domains $d_1, d_2, ..., d_n$.
- $F = \{f_1, f_2, ..., f_m\}$ is set of propositional formulas (rules or restrictions) over attribute set X.

In this paper the configuration variables X are boolean, hence domain $d_i \in \{0, 1\}, \forall i \in X$. A configuration is said to be feasible if an assignment for all attributes ($i \in X$) is found which fulfils each and every propositions in F. For configuration problem C a solution space $S(C)$ can be built by finding all assignment of configuration variable X which satisfy rules F. The problem we have in hand, the size of solution space $S(C)$ could be in the ranges of thousands of billion [6].

For a customizable product which changes with respect to time (due to introduction of new attributes, discontinuation of existing attributes or change in attributes dependencies) the configuration problem at any given time t can be expressed as $C_t = (X_t, D_t, F_t)$, where X_t,D_t and F_t are configuration variable, its domain and underlying propositional formulas respectively at time t. In this paper the domain D_t is fixed (boolean for all variables/attributes) so changes in configuration problem are possible by changing configuration variables X, changing rule set F or both.

In the scenario shown in Figure 1, let us assume that at time t the manufacturer wants to make some planning estimate for time $t + T$ (mid to long term planning, typically T = 6 months - 3 years) to support various planning activities such as production planning,

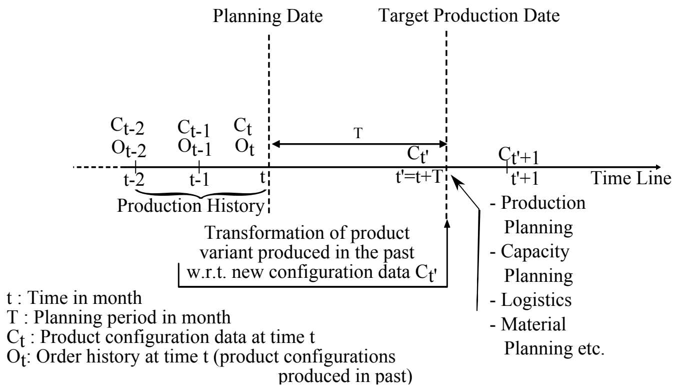


Figure 1: Product variant produced in the past need to be transformed w.r.t. new product design for use in future planning

capacity planning, material requirement, supplier selection. At time t the manufacturer has information about its current and past product configurations data ($C_t, C_{t-1}, ...$, capturing list of attributes/features (X) and its dependencies/rules (F)) and order history O_t, which is an 0-1 assignment of attributes. At any time t the validity of the product configurations will be checked according to rules written in $C_t = \{X_t, F_t\}$. As the product changes with respect to time, for every time instance we will have a corresponding product configuration problem instance. In practice, process of engineering change starts much before (typically 5-7 years) the start of production. This gives possibility to know the product configuration data for future time i.e. $C_{t+T} = \{X_{t+T}, F_{t+T}\}$ at given time t.

Now, for given set of configurations (O_t, will also be called base configuration) which are derived from configuration model $C_t = (X_t, F_t)$ we are required to validate their feasibility with respect to $C_{t+T} = (X_{t+T}, F_{t+T})$. In case of infeasible configurations we are required to find the new configuration in the solution space of $S(C_{t+T})$ with the minimal change to its base configuration. As the configurations variables are Boolean in nature, change in the configuration can be performed either by adding new attributes, or removing old attributes. The distance between two configurations (base and transformed) can be expressed through sum of the changes in the attribute assignment, which can be expressed through the Hamming distance. However, changing any arbitrary attribute in the base configuration in order to make them feasible may not be practically desired. For example, some of the product attributes may have high cost of change such as engine, special body style or sophisticated optional equipment, and changing these attributes may be difficult to handle as compared to changes in some simple options such as cup holder or some alarm features. Thus, a change cost can be associated with each attribute and transformation of base configuration to new configuration can be sought to be achieved by minimizing the total change cost. Change cost will only be associated to configuration if certain attribute is either added or removed in the configuration. One may consider two different quantities of change cost for an attribute such as attribute addition cost and attribute removal cost.

In case of *Reconfiguration* problem, some attributes are fixed by customer (attributes on which modification is asked) or may have very high change cost as they may be customer's most preferred attributes. Then the solution is sought only by changing the remaining set of attributes. The reconfiguration problem can be defined as a special case of configuration problem where certain configuration variables are set to predefined values (true or false). The aim is to fix certain attributes in base configuration (either by replacing some previously selected attribute or adding new) and then look for a new configuration which has minimal changes with respect to the base configuration.

In our case, the changes in the configuration can only be made either by adding new attributes or removing previously selected attributes from the configuration. As configuration changes are associated with change in attributes thus a change cost can be associated with each attribute to measure the impact of change.

In our work we propose an optimization model for transforming invalid configurations to valid ones as well as transforming configurations with predefined settings over attributes (Reconfiguration). We develop a framework which can incorporate information from different data sources such as configuration rules, sales program (cost associated with attributes) and planning expert's knowledge (to change configuration in some guided way). As most of the information is available or can be converted in the form of logical propositional formulas, we develop an optimization based framework after a required transformation of the logical propositions. In the next section we discuss various input data for the planning problem.

3 Input Data and its characteristics

3.1 The configuration data

A variant rich customizable product can be defined on the basis of attributes (features) in order to facilitate aggregate level of planning for components and modules [14]. Customer configurations can be created by combining different attributes that are permitted by the corresponding configuration data. It is important that while combining different attributes, we must fulfil the interdependencies between attributes, so that a feasible product configuration can be generated [13]. For instance, if in the USA some engines require special transmission types, this condition must hold while configuring a car of that type. A product document captures the technical, market and legal restrictions and provides an important data source for the configuration feasibility check.

Interdependencies among attributes are documented and maintained in the configuration data by a rule system. These rules are basically Boolean expressions imposed against each attribute. Selection of attributes in a configuration is done through evaluating the respective Boolean expression. Table 1 shows an example of such a data.

A customer configuration consists of a list of attributes. Each attribute is represented as a Boolean variable in the configuration data.

Attribute	Name	Rule	Description
1	Automatic climate control	$(2) \wedge (3 \vee 4)$	attribute 1 only when attribute 2 is present and either attribute 3 or 4 is present
2	Air condition	TRUE	must be present in every variant
3	Comfort package	$\neg(4)$	attribute 3 is not with attribute 4
4	Performance package	$\neg(3)$	attribute 4 is not with attribute 3

Table 1: Example: Rule based configuration data

The value of the attribute will be set to TRUE, if particular attribute is selected by the customer. The selection of the attribute is controlled by the logical rule system as shown in rule column of table 1. The logical rule system is built from usual Boolean operators $\vee$(OR), $\wedge$(AND), $\neg$(NOT) and an attribute serving as a proportional variable. The customer order processing is controlled by evaluating the rule's formulae under the variable assignment induced by the customer order and executing suitable actions based on whether the formula evaluates to TRUE or FALSE.

As discussed in section 2 configuration problem (C) can be defined by triple (X, D, F). For configuration data shown in table 1. $X = \{1, 2, 3, 4\}$,$D \in \{0, 1\} \forall X$, and $F = \{f_1, f_2, f_3, f_4\}$ where

$$f_1 = \{1 \rightarrow (2) \wedge (3 \vee 4)\}$$
$$f_2 = \{2\}$$
$$f_3 = \{3 \rightarrow \neg(4)\}$$
$$f_4 = \{4 \rightarrow \neg(3)\}$$

where $a \rightarrow b$ means attribute a implies attribute b, if a is selected (or set to true) then b has to be selected in the configuration. Propositional formulas in F can also be expressed as $F = \{((2) \wedge (3 \vee 4)) \vee \neg(1), 2, \neg(4) \vee \neg(3), \neg(3) \vee \neg(4)\}$. In the given example, associated rule with f_3 and f_4 have the same boolean expression so only one can be evaluated and also $f_2 = \{2\}$ says that attribute 2 will be the part of every configuration. As all rules written in the configuration rule set F has to be satisfied. All element of F can be combined with AND operator, $\varphi = \wedge_{f \in F} f$. Thus φ will be the boolean formula whose Truth value will represent an configuration. φ is also called as *product overview formula* [9]. Our configuration variable set X contains all possible attributes which can be the part of the product configuration either from customer point of view of manufacturer. For example, some plant and production related attribute may not be relevant to the customer but is required to handle feasibility of production at certain planning stage. In the next section we discuss different changes in the configuration data which may result in modification or upgradation of configurations.

3.2 Changes in the configurations

As a customizable product can be defined based on different features offered by the manufacturer, product changes can be studied based on the change in the offered product attributes. In this section we will outline various changes in product attributes which can make certain product variants to invalid. The changes in the product attributes can be cause by one or more of reasons described below:

1. Deletion of old attributes: All past configurations containing attributes which are discontinued will become invalid according to changed product. If discontinued attributes have no dependencies with remaining attributes we can simply remove these attributes to restore the validity (feasibility) of the product variant. For a complex engineering product this is very unlikely. In general, product attributes have complex dependencies among each other and modification of one attribute needs to be validated with the remaining set of attributes.
2. Change in rule: The technical rules pertaining to an attribute that are expressed in configuration data may get changed due to various reasons such as design modification, legal changes. For some practical product instances, a single attribute may depend on hundreds of other attributes by a complex Boolean expression. Change in some part of a rule may affect feasibility of certain attribute combinations.
3. Inclusion of new attributes: As a product evolves, some new features get added. These may not have been present in the past, but a customer may select them in the future. As newly introduced features may have some dependencies with other available attributes, variants produced in the past have to be modified in such a way, that transformed configurations also contain new features (according the estimate of new feature).
4. Attribute fragmentation/atomization: In some cases an attribute is split into more attributes. For example, let us assume that a car was produced with the option off-road package which includes features as high battery capacity, heavy duty suspension, hi-fi music system and a sunroof. Customers were not allowed to select above features individually but selection can be made through package. Now, due to some change, the manufacturer has decided to divide the off-road package into two new packages. The first package includes the features high battery capacity and heavy duty suspension, the second package includes the hi-fi music system and sunroof. Both new packages can be selected individually, which means that the customer has more choice than before which may effect the distribution of packages from the past. Some input form sales in-terms of demand estimates of new package may help here to adapt past configurations according to new product offerings.
5. Replacement of attributes: Most often due to technology and other changes, some old attributes are replaced by new attributes. For example, some old telematic features are replaced by the new generation touchscreen based systems. Therefore historic product variants should also be upgraded to the new generation to use them for planning of a future production system.

Apart from the above changes there also exists some desire to change attributes of a past product variant according to new product offerings. For example, due to market changes, the demand for a certain engine type may decrease in comparison to other available engines. In this case the changes in the engine distribution across all transformed historic orders have to be considered in the transformation process. This information is not documented in the configuration rules but can be accessible through planning experts or through some sales forecast. During the development of the automatic configuration transformation system we try to accommodate these kinds of requests.

3.3 Customer history

Let a product be defined by a set of 4 attributes $\{1, 2, 3, 4\}$. According to Table 1, the configuration can be listed as described in Table 2 and 3. As shown in Table 3, customer configuration can be presented as a 0/1 vector over attributes, any change in the configuration can be made by changing attributes from 0 to 1 and vice-versa. While transforming the configuration, one objective will be to be as close as possible to the old configuration. This can be done by minimizing the Hamming distance between the old and new configurations.

No.	**Configurations**
i	1,2,3
ii	1,2,4
iii	2,3
iv	2,4

Table 2: Configuration based on attributes set

No.	**1**	**2**	**3**	**4**
i	1	1	1	0
ii	1	1	0	1
iii	0	1	1	0
iv	0	1	0	1

Table 3: Configuration as 0/1 matrix over attributes

4 Formulation of the optimization model

During the transformation of product configurations, we need to evaluate each rule written in the corresponding configuration data. At the same time, we also need to ensure, that the changes in the given product variant are done with minimal cost. Cost can vary based on deviation from the base configuration and the type of changes done. In this section we explore an optimization based framework to find a solution for the above problem. To create an optimization based transformation procedure, all information included in the product configuration process need to be considered in the model. To do this, in the following section we first transform rules from the configuration data to the corresponding 0-1 discrete programming equivalent forms.

4.1 Transformation of logical rules to linear inequalities

Constraint programming approach is a well-used methodology inside the many product configuration systems [2]. Restrictions on product configurations are modelled as constraints and a solution is a total assignment satisfying each of the constraints. Most of the proposed framework rely of the transformation of boolean formulas to special structure such as conjunctive normal form (CNF) before writing the final constraint set [3]. We developed an alternate method to avoid the initial conversion of the input to CNF. Our formulas are so large that naive CNF conversion by applying the distributive law failed for lack of memory and time. Also, CNF conversion steps involves introduction of large number of new variables which increases the complexity of the problem.

4.1.1 Data structure for configuration rules

Using the normal precedence operators and the conventional evaluation of expressions, the logical rule from configuration data (F) can be presented in form of a tree structure. For example, let's say selection of an attribute 1 is controlled by following Boolean expression:

$$f_1 = (2) \wedge (3 \vee 4) \tag{1}$$

The tree representation of above expression can be shown as Figure 2. We used *Stack* for storing binary tree for implementation of algorithm for transforming logical rules to algebraic inequalities [12].

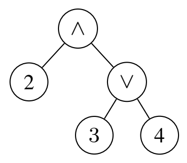


Figure 2: Representation of attribute selection rule in a binary tree

index	Elements
0	∧
1	2
2	∨
3	3
4	4

Figure 3: Rule in a stack

4.1.2 Transforming propositional formula's to 0-1 LP

In this section we describe the transformation of logical propositions to its equivalent linear integer constraint through an example. The procedure to obtained required transformation is discussed in [12] and [1]. Linear inequalities over Boolean variables are a widely used modelling technique. The main task during transformation of an attribute selection rule into a system of linear constraints is to maintain the logical equivalence of the transformed expressions. The resulting system of constraints must have the same truth table as the original statement. For every attribute we introduce a binary decision variable, denoted by x_i. The connection of these variables to the propositions is defined by the following relations:

$$x_i = \begin{cases} 1 & \text{iff attribute i is TRUE} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Imposition of logical conditions linking the different actions in a model is achieved by expressing these conditions in the form of linear constraints connecting the associated decision variables.

Let us assume that a product is defined by five different attributes as shown in table 4. Our task is to write a set of linear constraints which represents same information as described for configuration

Attribute	Name	Selection Rule
1	Rear-view camera	$1 \rightarrow \neg(4 \vee 5) \wedge (\neg 6)$
2	Parking assistant system	$2 \rightarrow (1) \wedge (\neg(4 \vee 5))$
3	Cruise control	$3 \leftarrow (1 \vee (4 \wedge 5))$

Table 4: Example: attributes and their selection rule

problem. In this example attribute 1, 2 and 3 imposes a selection rule criteria while attribute 4 and 5 do not have explicit dependencies.

Our approach, in principle, involves identification of precise compound attribute rules of the problem and then processing it with identified equations. The logical rule is represented by a tree graph (as per Section 4.1.1), where attributes are associated with their common operator node. We traverse through the tree and prune it in such a way, that the standard transformation equation can be applied [12]. Figure 4 shows the final expression tree for configurations rule written in Table 4.

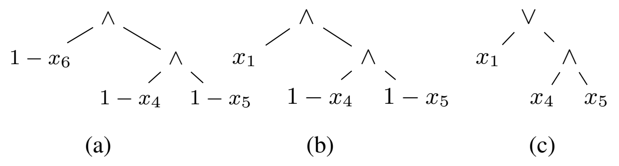


Figure 4: Example: Final expression tree for (a) Attribute 1 (b) Attribute 2 (c) Attribute 3

$$\begin{bmatrix} 3 & 0 & 0 & 1 & 1 & 2 & 0 \\ -1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & -2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \leq \begin{bmatrix} 4 \\ 2 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{B} \times [x] \leq \mathbf{b} \tag{3}$$

Where: B = Coefficient matrix over attributes and b is the right-hand side values. In order to transform the given Boolean expressions to liner constraints we introduced new variable x_7 corresponding to attribute 3. Attribute x_7 controls boolean expression $4 \vee 5$. Resulting constraint system is shown in Eq. 3.

4.2 The configuration transformation model

In this section we present a mathematical model for the transformation of a base configuration (configuration produced in past) to the new configuration. The new configuration should satisfy all restrictions imposed by product document and should have maximum correlation with its base configuration.

Let
i be i^{th} attribute, $i \subseteq \{1...n\}$, where n is the total number of attributes

Data

$$a_i = \begin{cases} 1 & \text{if } i^{th} \text{ attribute is present in base configuration} \\ 0 & \text{otherwise} \end{cases}$$

$c_i =$ Change cost associated with attribute i. We assume that c_i is given as input either from user or derived from sales planning data (e.g. cost of attribute)

Decision variables

$$x_i = \begin{cases} 1 & \text{if } i^{th} \text{ attribute is in transformed configuration} \\ 0 & \text{otherwise} \end{cases}$$

Objective Function

$$\textbf{Z = Minimize} \sum_i c_i \times |a_i - x_i| \tag{4}$$

Subject to

$$B[x] \leq b \tag{5}$$

The Hamming distance between base and new configuration for attribute i calculated by $|a_i - x_i|$. Constraints in Eq. 5 is the set of linear inequalities derived from configurations rules (restrictions). Objective function Z is used to minimize the mismatch cost associated with each attribute so that the transformed configuration will match the base (old) configuration as close as possible. Change cost associated with each attribute is assumed here as an input data provided either by planning experts (sales) or by user. Usually for automobile change cost for complex attributes such as power train, production country is high compare to other attributes. In this case user can specify relative cost (such as weight factor or priorities) among attributes. Constraint 5 is a set of linear constraints originating by transforming logical conditions written in the product document to linear inequalities using the procedure described in Section 4.1. Any new configuration $[x]$ from the above optimization model will guarantee that the configuration is feasible according to the product document and the objective function will ensure its minimum cost deviation from the base configuration. As the configuration transformation model transforms one configuration at a time, for every transformation of non-feasible (according to given product document) configuration, this model needs to be run. A typical practical instance of this problem contains around 500-1000 decision variables and some tens of thousands of constrains.

5 Solution framework

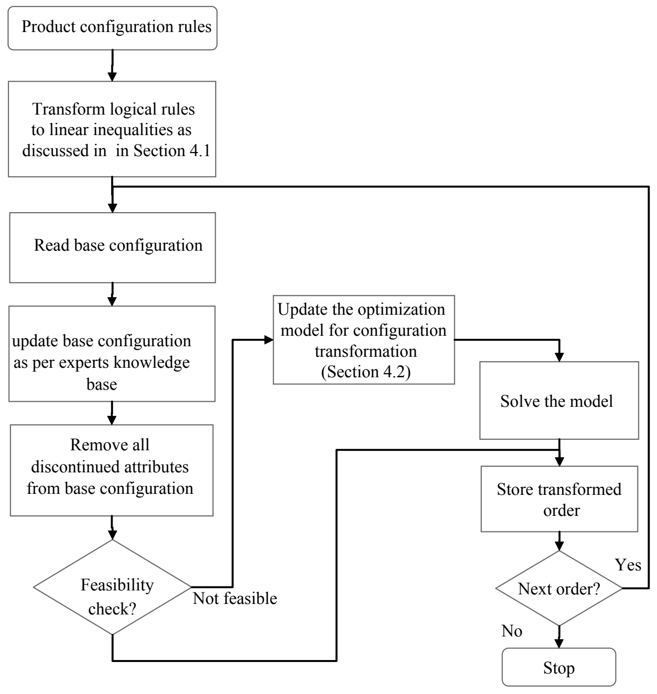
Our aim is to provide an automated system which can interpret information from configuration data and planning experts. The system should consider given information in the best possible way while transforming the base (given) product variants to new (upgraded) variants. For this, we will create a knowledge database, where information from planning experts can be stored and used during the configuration transformation. The term *planning experts* is used to present collective information/rules specified by engineers/product managers or the user of the our application. For the reconfiguration problem, change information can be described by the customer and same can be applied during updating the base configuration. The expert database will collect the changes of attributes from one stage of product to another. Table 5 shows an excerpt of such a knowledge database. In the expert database we want to maintain an explicit set of rules which can be applied in a guided way to base configuration. For example, in the past, a car was produced only with one type of entertainment system. Due to some enhancement in the product, the manufacturer now provides three different entertainment systems. The challenge will be to distribute new entertainment systems over configurations produced in the past. In this case, the knowledge of the planning expert plays an important role in achieving a realistic transformation of past products.

Situation	**in Past**	**in Future**
An old attribute is replaced by new attribute (one to one mapping)	i	j
An attribute has been replaced by number of new attributes (one to many mapping)	i	$i = j$ for 70% configurations produced in past; $i = k$ for 30% configurations
Group of individual attribute replaced by new Package (many to one mapping)	i, j, k	$p = [i, j, k]$ add package p if at least two attribute from $\{i, j, k\}$ is present
An old package is divided in more than one packages	$p = [i, j, k, l]$	$p_1 = [i, j]$, $p_2 = [k, l]$

Table 5: An excerpt of expert's knowledge database

Knowledge from the expert database is applied to every configuration that we want to transform. It may happen that the modifications from the expert database do not suffice to meet all the configuration rules. In that case, we use the configuration transformation model presented in section 4.2. A solution is sought automatically that is valid under the new model, but which differs minimally (in the "Hamming distance") from the "old" configuration. The flow diagram in Figure 5 shows the solution framework.

The configuration transformation process starts with analyzing the product configuration rules. In this step, we can get the list of all available attributes and attribute dependencies in terms of logical rules. These rules can be converted into a set of linear inequalities as discussed in Section 4.1. Once the configuration rules are modelled as constraints, we will look into the expert database and apply all possible attribute mappings described in the expert database. All discontinued attributes will be removed from the base configuration because they will not be valid for the new model. At this stage, we will check if this configuration is feasible according to given configuration rules. If the answer is YES, we proceed with transforming the next configuration. If the answer is NO, we call the configuration transformation model as defined in Section 4.2. We repeat the above procedure till all configurations are transformed.



Product configuration rules
Transform logical rules to linear inequalities as discussed in in Section 4.1
Read base configuration
update base configuration as per experts knowledge base
Remove all discontinued attributes from base configuration
Feasibility check?
Not feasible
Update the optimization model for configuration transformation (Section 4.2)
Solve the model
Store transformed order
Next order?
Yes
No
Stop

Figure 5: Flow diagram for transformation of product variants from past to given document information

5.1 Computational Experiments

We have tested our solution approach with various industry size problems. In this section, we will present two different experiments created out of practical scenarios in the automotive industry.

Sr	Scenario	total attributes	total base configurations
Exp_1	Transforming past configurations as per changes in configuration rules	695	2200
Exp_2	Upgrading base configurations with new Engine	705	1000

Table 6: Excerpt of computational scenarios

Table 6 shows the computational set up for two experiments. In the first experiment (Exp_1) our aim is to utilize customer orders produced in the past for future production planning. For this, 2200 past configurations are taken which are 6 months old from new production date. As the product has undergone engineering changes, our aim is to upgrade the given configurations as per the new configuration rules. The new configurations are defined with 695 different attributes.

Figure 6 shows the plot of time versus Hamming distance for transformed configurations. The transformation is done after analysing new configurations rules which results in information such

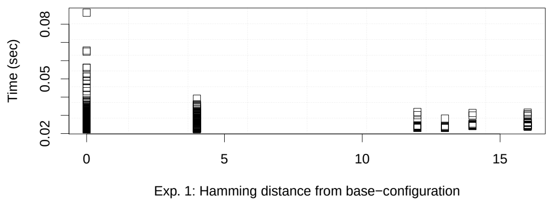


Figure 6: $Experiment_1$ Transforming base configurations as per new configuration rules

as discontinuation of some old attributes. Removing of barred attributes and application of information from expert's knowledge as discussed in section 5, we found that a large number of configurations become feasible (Hamming distance zero in figure 6) . For other configurations, solutions are found by solving the optimization model as discussed in section 4.

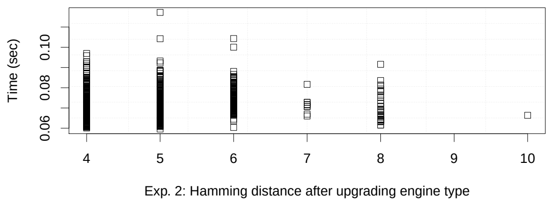


Figure 7: $Experiment_2$ Hamming distance vs time plot of engine upgradation problem

In experiment 2, we solved the reconfiguration problem by upgrading the engine type. Given 1000 configurations were upgraded to a new engine type. First, attributes related to the old engine type were replaced with the new engine and some related attribute replacements were done through expert's knowledge base. For example, associating the right gear box for the new engine. After user's modification, we transformed the given configurations as per the model shown in Figure 7. A large number of given configurations are transformed with minimal changes (4-8 attributes) to its original values. The optimization model out of configuration rules has a few thousand decision variables and thousands of linear constraints.

We used the optimization solver IBM Ilog Cplex 12.2 to solve the order transformation model. For simplicity, the following assumptions were made: 1) the attribute change cost is assumed to be one in $Experiment_1$. 2) In $Experiment_2$ we used a relatively high change cost for new engine and in all transformed configurations, the attributes related to new engine remained unchanged. On applying expert knowledge and the mathematical model that we have developed, the initial computational results shows that the given configurations can be transformed as per the desired objective in reasonable computation time (a few seconds).

6 Related work

Product configuration systems have been a key enabler for mass customization. One main contribution of configurations system is to support mass customization at various key processes such as product configuration, product data management (PDM) and customer relationship management (CRM) for effective product and process variety management [5]. The effect of configuration process can be seen on the customization responsiveness when information from sources such as customer requirements, product characteristics, production process and logistics network are considered in the configuration systems [8].

In a variant rich customizable product, finding customer focused configurations out of enormous choices is a challenging task [16]. Failing to access market needs has an adverse effect in product quality of product configurators [15]. Enabling production planning with customer historical demand (configurations produced in the past) may help to retain aspects of customer buying behaviour. However, to use past configurations for future production planning, an upgradation is required. Fichter et. al. [4] considered some of the product change conditions in their work of transforming configurations between two different product document rules. They proposed a knowledge based framework to transform invalid product variants according to change of rules in a configurator. However, in their heuristic approach it is not clear whether the transformed configuration has small deviation (minimal cost/distance) from original configuration. Walter et. al. [17] have discussed MaxSAT based approach for reconfiguration problem. In our paper we translated configurations propositional rules to set of linear constraints and the configuration transformation problem. An optimization based model has some advantages and the results of this formulation can be extended to support the generation or the transformation of sets of configurations [12].

7 Conclusion

In order to adapt the customer configurations produced in the past to the latest engineering design and market conditions we have discussed an optimization based framework. Design related changes are captured in our optimization model by transforming the product configuration rules to a set of linear inequalities. Market and expert knowledge during configuration transformation are captured by maintaining a knowledge database to transform configurations according to the best available information. The method will facilitate future planning activities based on consistent and constructible configuration sets (order sets), which will have maximum correlation with the past customer demand. For a complex product which changes dynamically with respect to time, production planning activities will improve gradually with the effective adaption of design and market changes.

REFERENCES

[1] Egon Balas, ‘Logical constraints as cardinality rules: Tight representation’, *Journal of Combinatorial Optimization*, **8**(2), 115–128, (2004).

[2] Caroline Becker and Hélène Fargier, ‘Maintaining alternative values in constraint-based configuration’, in *IJCAI*, ed., Francesca Rossi. IJCAI/AAAI, (2013).

[3] Hachemi Bennaceur, ‘A comparison between sat and csp techniques’, *Constraints*, **9**(2), 123–138, (2004).

[4] Michael Fichter, Michael Klein, and Andreas Schmidt, ‘Transformation of product between various version of the rule world of a product configurator’, *IEA/AIE, Springer-Verlag Berliun Heidelberg*, **5579**(1), 721 – 730, (2009).

[5] C. Forza and F. Salvador, ‘Application support to product variety management’, *International Journal of Production Research*, **46**(3), 817–836, (2008).

[6] H. Graf, ‘Innovative logistics is a vital part of transformable factories in the automotive industry’, in *Reconfigurable Manufacturing Systems and Transformable Factories*, ed., AnatoliI. Dashchenko, 423–457, Springer Berlin Heidelberg, (2006).

[7] T. Hadzic, S. Sathiamoorthy, R. M. Jensen, H. R. Andersen, J. Møller, and H. Hulgaard, ‘Fast backtrack free product configuration using precompiled solution space representations’, in *Proceedings of the International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems*, (2004).

[8] P.T. Helo, Q.L. Xu, S.J. Kyllnen, and R.J. Jiao, ‘Integrated vehicle configuration systemconnecting the domains of mass customization’, *Computers in Industry*, **61**(1), 44 – 52, (2010).

[9] Wolfgang Küchlin and Carsten Sinz, ‘Proving consistency assertions for automotive product data management’, *Journal of Automated Reasoning*, **24**(1-2), 145–163, (2000).

[10] Andrew Kusiak, M. R. Smith, and Zhe Song, ‘Planning product configurations based on sales data’, *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, **37**(4), 602–609, (2007).

[11] Peter Manhart, ‘Reconfiguration - A problem in search of solutions’, in *IJCAI’05 Configuration Workshop*, eds., Dietmar Jannach and Alexander Felfernig, pp. 64–67, (2005).

[12] Tilak Raj Singh and Narayan Rangaraj, ‘Generation of predictive configurations for production planning’, in *15 th International Configuration Workshop*, p. 79, (2013).

[13] C. Sinz, A. Kaiser, and W. Küchlin, ‘Formal methods for the validation of automotive product configuration data’, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **17**(1), 75–97, (JAN 2003). Special issue on configuration.

[14] R. Srinivasan and J. M. Swaminathan, ‘Managing configurable products in the computer industry: Planning and coordination issues’, volume 22, pp. 33–43. Sadhna:Academy Proceedings in Engineering Sciences, (February 1997).

[15] Alessio Trentin, Elisa Perin, and Cipriano Forza, ‘Product configurator impact on product quality’, *International Journal of Production Economics*, **135**(2), 850 – 859, (2012). Green Manufacturing and Distribution in the Fashion and Apparel Industries.

[16] Alessio Trentin, Elisa Perin, and Cipriano Forza, ‘Sales configurator capabilities to avoid the product variety paradox: Construct development and validation’, *Computers in Industry*, **64**(4), 436 – 447, (2013).

[17] Rouven Walter, Christoph Zengler, and Wolfgang Küchlin, ‘Applications of maxsat in automotive configuration’, in *15 th International Configuration Workshop*, volume 1, p. 21, (2013).