# A WEIGHTED SET COVER PROBLEM FOR PRODUCT FAMILY DESIGN TO MAXIMIZE THE COMMONALITY OF PRODUCTS

**Han, Hyeongmin (1); Chang, Sehyun (2); Kim, Harrison (1)**

1: University of Illinois at Urbana-Champaign; 2: Hyundai Motor Company

## ABSTRACT

In product family design, the commonality of products and performance are competing objectives when designers build platforms. The commonality makes it efficient to manufacture products while it will cause performance loss of products. In this paper, we assume that performance functions evaluate the performance of a product. Targets of performance functions are set for each product depending on the product's property. The designs that satisfy the target of performance functions are denoted as 'good' design points. By using 'good' design points, a weighted set cover problem (WSC) is applied to formulate the combinatorial optimization problem, which maximizes the commonality by minimizing the number component attributes. A recursive greedy algorithm is proposed to handle the general cost function in the problem for product family design. The formulation and the algorithm are tested for a linear three-degree-of-freedom (3DOF) model. In numerical experiment, the proposed method determines optimal values of the components which are suspensions, stabilizer bars, and tires in the vehicle model.

**Keywords**: Product families, Platform strategies, Optimisation

**Contact**:
Kim, Harrison
University of Illinois at Urbana-Champaign
Industrial and Enterprise Systems Engineering
United States of America
hmkim@uiuc.edu

**Cite this article:** Han, H., Chang, S., Kim, H. (2019) 'A Weighted Set Cover Problem for Product Family Design to Maximize the Commonality of Products', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.302

# 1 INTRODUCTION

Product family design provides efficient product platforms that are used to produce similar types of products. The variety of products enables manufacturers to meet the needs of various markets. Also, platform-based products reduce costs and make logistics effective (Simpson, 2004). However, building a platform for different types of products can cause performance loss. Also, commonality of the platform design will affect the modularity in each product. In the early stage of the product family design, designers have to consider the performance of products, commonality of components, and modularity of an individual product. Scale-based product family design (Simpson *et al.*, 2001) and module-based product family design (Du *et al.*, 2001) can be used to create new product platform from the products that are produced from existing platforms. The scale-based product family design scales the platform to create variants of the product family, while module-based product family design generates products by substituting components.

When the designers build platform design, the key factor of product family design is the efficiency of the platform. The commonality among the variants would reduce the manufacturing cost of the products which eventually increase the net profit. Also, designers should consider the performance loss caused by sharing components. The main goal of the optimizing the platform design is maximizing the commonality while minimizing the performance loss. Various approaches that optimize the product family platform have been proposed. In the early studies, sharing components are selected prior to designing the optimal components. Nelson *et al.* (2001) proposed a multi-criteria optimization to optimize components when the sharing components are given. The result is compared to the optimal design without sharing so that they can quantify the performance loss caused by the commonality. Gonzalez-Zugasti *et al.* (2000) built a multidisciplinary optimization problem for platform selection. They proposed an interactive approach to design the product family. Platform design processes and variant design processes are done iteratively until the platform design team and the variant design team satisfy with the results from each other. Rai and Allada (2003) optimized the combination of components under the predefined component candidates. Also, Rojas Arciniegas and Kim (2011) provided an iterative process that optimizes the modularity and commonality of product families with predefined component candidates.

There are several methods in the literature that optimize the component selection and component design simultaneously. Fujita and Yoshida (2004) developed an optimization problem that maximizes the profit under performance constraints. They formulated a problem that optimizes the module attributes and selection simultaneously. The problem is solved by a genetic algorithm. Simpson and D'souza (2004) also presented a model with all-common or all-distinct restriction. That means a component should be either in one product or all products, but partial sharing is not allowed. The limitation had been relaxed by Khajavirad *et al.* (2009). The optimization problem is formulated as bi-level decomposition problems. Chowdhury *et al.* (2011) used a similar commonality matrix to formulate a mixed integer non-linear programming (MINLP). They maximized the performance of each product and minimized the production cost.

In product family design, performance functions of products are used to evaluate the performance loss and maximize the performance of the product. For complex product designs such as vehicle design and aircraft design, numerically expensive performance functions are handled as those includes dynamics and simulations. Due to their complexity, designers can set a target range for each performance function and define the acceptable region of design variables. The region is called a solution space. Designs in the solution space are identical in terms of the performance as they satisfy the target range. Instead of evaluating the performance functions for every iteration, a predetermined solution space can be used in optimization processes. Also, designers will get an infinite set of solutions so that they can easily adapt to a new target rather than having one best point. Eichstetter *et al.* (2015) set target ranges for performance functions as constraints and analyzed the solution spaces. Good design points that are in the solution space are sampled with Monte Carlo simulation and Bayesian statistics. They estimated the solution space with a box-shaped region. Since the estimation is a Cartesian product of the lower and upper bound of each design variable, the method has limitations if the actual solution space is nonconvex or has disconnected sets.


https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

In this paper, we used a finite set of the solution space for each product. Since the actual solution space is computationally expensive to be obtained unless it is an approximation, discrete design points are utilized in the proposed method. With the good design points in the finite set, a combinatorial optimization problem is solved to maximize the commonality of the product family design. In section 2, the proposed method is introduced. In section 3, a numerical experiment is performed for a vehicle design problem. Lastly, discussion and conclusion are provided in section 4 and 5.

## 2 METHOD: MAXIMIZING COMMONALITY WITH WEIGHTED SET COVER PROBLEM

### 2.1 Overview

In the early stage of a platform design, target ranges of performance functions determine the characteristics of different products in the product family. For example, the target range can aim higher performance level for a high-end product than for a low-end product. If there are $P$ number of products to be designed, each of the product has different boundaries of design variables and target ranges of performance functions. The solution space for each product $p$ can be represented as follows.

$$\mathscr{F}_p = \{\mathbf{x} \mid \mathbf{x_p^L} \leq \mathbf{x} \leq \mathbf{x_p^U}, \mathbf{T_p^L} \leq \mathbf{R}(\mathbf{x}) \leq \mathbf{T_p^U}\} \quad \forall p = 1, ..., P \tag{1}$$

In Equation 1, $\mathbf{x}$ is a vector of design variables which define attributes of components. If the values are different, components are considered as distinct components. A lower bound and upper bound of $\mathbf{x}$ are denoted as $\mathbf{x_p^L}$ and $\mathbf{x_p^U}$. $\mathbf{R}(\mathbf{x})$ represents the performance function, and $\mathbf{T_p^L}$ and $\mathbf{T_p^U}$ are target ranges.
The main goal of the proposed method is maximizing commonality of product family by minimizing the number of distinct components. The method starts with the table of candidate designs. An example of the table is shown in Table 1 where $S$, $N$, and $P$ represent the number of candidate design point, design variables, and products, respectively. In the table, $\theta$ represents an attribute value of the component. For the $n$-th design variable, $a_n$ means the number of distinct attribute values for the variable. Each row of the table contains information about the design point. For example, design 1 consists of the components $\theta_1^1$, $\theta_2^1$, and $\theta_N^1$ for the design variables 1, 2, and $N$. As it is denoted as 'Yes' for the product 1 and there are 'No' for the rest of products, design 1 satisfies the boundary and the target range only for product 1. In the table, design 1, design 2, and design 3 shares the component $\theta_1^1$ for design variable 1. Also, design 2 and design 3 satisfy multiple products. As the table is a collection of candidate design points, the designs that do not satisfy any product are not included in the table.

*Table 1. A table of candidate designs*

| | Design variables | | | | Products | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | ⋯ | $N$ | 1 | 2 | ⋯ | $P$ |
| Design 1 | $\theta_1^1$ | $\theta_2^1$ | ⋯ | $\theta_N^1$ | Yes | No | ⋯ | No |
| Design 2 | $\theta_1^1$ | $\theta_2^2$ | ⋯ | $\theta_N^2$ | Yes | Yes | ⋯ | No |
| Design 3 | $\theta_1^1$ | $\theta_2^2$ | ⋯ | $\theta_N^3$ | Yes | No | ⋯ | Yes |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| Design $S$ | $\theta_1^{a_1}$ | $\theta_1^{a_2}$ | ⋯ | $\theta_N^{a_N}$ | No | No | ⋯ | Yes |

Designers will choose design points that will satisfy all the product in the family while minimizing the manufacturing cost. It is a combinatorial optimization problem with a substantial number of instances (design points). The objective of the problem is minimizing the number of distinct components, which will minimize the cost. The constraint is that the selected design points have to satisfy the boundaries and target ranges for the product family. In the table, for every product, at least one selected point should be 'Yes' for the product. The proposed method utilizes the table to formulate the product family design as a weighted set cover problem (WSC). Also, we introduce a recursive greedy algorithm to solve the problem.
The table of candidate designs can be obtained from the earlier version of the products. If designers don't have enough data, they have to do preprocessing to creates the table. The detailed process is explained in the following subsection.


https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

## 2.2 Preprocessing

In this section, the preprocessing that provides the table of candidate designs is explained. We assumed that performance functions and the target range are given for each product and the attributes are continuous values. Designers can predict the solution space in the design variable space and sample design points that satisfy the boundaries and at least one target range of the performance function. With an existing sampling technique such as Monte Carlo simulation, the sampling process starts from $\mathscr{F}_1$, which is the solution space of the first product. For the samples in $\mathscr{F}_1$, we test their feasibility for the other solution spaces ($\mathscr{F}_2$, ..., $\mathscr{F}_N$). When the sampling process for the first product is done, the same process is performed for the second product and so on. Collected points are denoted as $\mathbf{x}^s = (x_1^s, x_2^s, ..., x_N^s)$, where $s$ is from 1 to $S$. For each design sample, attributes of the components and the feasibility for the products are recorded in the table.

As the attributes are continuous values, all the components in the samples are distinct. For each design variable, the tolerance of attribute are defined as $\epsilon = (\epsilon_1, \epsilon_2, ..., \epsilon_N)$. The element in $\epsilon$ defines the size of a grid for each variable and discretize the continuous design variables. For design variable $n$ and product $p$, the number of component attributes that the variable can have is $\lceil x_{p,n}^U - x_{p,n}^L / \epsilon_n \rceil$. By discretizing the design variables, we can redefine the attributes of the samples as $\bar{x}_n^s$ which is a discretized value of $x_n^s$. Since the values become discrete, there are some duplicated values of $\bar{x}_n^s$ in the table of candidate designs. For example, there are two products ($P = 2$), two design variables ($N = 2$), and three samples ($S = 3$). The sample points are $\mathbf{x}^1 = (3.54, 7.61)$, $\mathbf{x}^2 = (3.52, 8.47)$, and $\mathbf{x}^3 = (3.35, 8.29)$. In this example, $\mathbf{x}^1$ satisfies the first product, $\mathbf{x}^2$ satisfies the first and second product, and $\mathbf{x}^3$ satisfies the second product. The sizes of grids are defined as $\epsilon = (0.1, 0.5)$. Then the discretized points are $\bar{\mathbf{x}}^1 = (3.5, 7.5)$, $\bar{\mathbf{x}}^2 = (3.5, 8.0)$, and $\bar{\mathbf{x}}^3 = (3.3, 8.0)$. For the discretized points, two equations are obtained; $\bar{x}_1^1 = \bar{x}_1^2$ and $\bar{x}_2^2 = \bar{x}_2^3$. After reordering the indices, the table of the candidate design for the example is following.

*Table 2. An example of candidate designs*

| | Design variables | | Products | |
|---|---|---|---|---|
| | 1 | 2 | 1 | 2 |
| Design 1 | $\theta_1^1 = 3.5$ | $\theta_2^1 = 7.5$ | Yes | No |
| Design 2 | $\theta_1^1 = 3.5$ | $\theta_2^2 = 8.0$ | Yes | Yes |
| Design 3 | $\theta_1^2 = 3.3$ | $\theta_2^2 = 8.0$ | No | Yes |

## 2.3 Formulation: Weighted set cover problem (WSC)

WSC is a classical problem in graph theory. Given a set of elements $\mathscr{U}$, subsets, $\mathscr{S}$ ($\subseteq \mathscr{U}$), are selected to cover all the elements while minimizing the cost, $\mathbf{c}$. In the problem, there is a fixed cost for each subset. Since the weighted set covering problem is NP-complete, a greedy algorithm is used as a polytime approximation (Chvatal, 1979). The greedy algorithm evaluates the ratio of the cost and the number of added elements. For each iteration, a subset that has a minimum ratio is added to the solution. The iteration ends when selected subsets cover all the elements. The greedy algorithm is presented in Algorithm 1.

**Algorithm 1** Greedy algorithm for WSC

**Input:** Universe ($\mathscr{U}$), Subsets ($\mathscr{S}$), Cost ($\mathbf{c}$)
**procedure** GREEDY($\mathscr{U}$, $\mathscr{S}$, $\mathbf{c}$)
  $\mathscr{I} \leftarrow \emptyset$
  $\mathscr{X} \leftarrow \emptyset$
  **while** $\mathscr{I} \neq \mathscr{U}$ **do**
    Pick $\mathscr{S}^* \in \mathscr{S}$ that minimizes $\frac{c_{\mathscr{S}^*}}{|\mathscr{S}^* - \mathscr{I}|}$
    $\mathscr{X} \leftarrow \mathscr{X} \cup \{\mathscr{S}^*\}$
    $\mathscr{I} \leftarrow \mathscr{I} \cup \mathscr{S}^*$
  **end while**
  **return** $\mathscr{X}$
**end procedure**



https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

We formulate the maximum commonality problem as an weighted set cover problem. In the product family design, the products become elements, and the design points represent subsets of the WSC problem. Each subset has product number if the corresponding design point satisfies the product. For instance, the subset $\mathscr{S}_7 = \{1,3\}$ means the seventh design point satisfies product 1 and 3. The cost represents the total number of components that are used.

$$\begin{aligned} &\text{Universe:} \quad \mathscr{U} = \{1,2,...,P\} \\ &\text{Subsets:} \quad \mathscr{S} = \{\mathscr{S}_1, \mathscr{S}_2, ..., \mathscr{S}_S\} \\ &\text{Cost:} \quad \mathbf{c}(\mathscr{S}) \end{aligned} \tag{2}$$

The problem has two main features that are different from the traditional WSC problem. Because of the difference, the recursive greedy algorithm is proposed by modifying the original greedy algorithm. The proposed algorithm is shown in Algorithm 2. First, the costs are defined as functions of all the subsets, $\mathbf{c}(\mathscr{S})$. In WSC, the cost function only depends on one subset and does not consider interactions among subsets. However, in our formulation, we take into account all the interactions. For example, there are three design points in total ($\bar{\mathbf{x}}^1 = (\theta_1^1, \theta_2^1, \theta_3^1, \theta_4^1)$, $\bar{\mathbf{x}}^2 = (\theta_1^1, \theta_2^2, \theta_3^2, \theta_4^1)$, and $\bar{\mathbf{x}}^3 = (\theta_1^1, \theta_2^2, \theta_3^3, \theta_4^2)$). As we select a point iteratively by the greedy algorithm, the total cost increases as iteration continue. Let's say the first design point, $\bar{\mathbf{x}}^1$ is selected. The cost becomes 4 as we added four different components ($\theta_1^1$, $\theta_2^1$, $\theta_3^1$, and $\theta_4^1$). In the second iteration, if $\bar{\mathbf{x}}^3$ is selected, the total cost is 7 because three new components added as the first component, $\theta_1^1$, is already used by $\bar{\mathbf{x}}^1$. As the cost function in the formulation is different from the traditional WSC, the greedy algorithm cannot be directly applied. In the proposed method, the cost is set as $N$ initially and updated for every iteration. The maximum cost of adding one subset (design point) is $N$ if there is no sharing component in the design point. In order to update the cost function, components that are used by the selected design points are defined as active components.For each unselected design point, if $b$ is the number of components that are in the set of active components and used for the selected points, the cost of the design point become $N - b$. The updating is done before evaluating the minimum ratio.

**Algorithm 2** Recursive greedy for maximum commonality

```
Input: Products (𝒰), Points (𝒮), Cost (c),Candidate points (𝒮^LIST), Selected points (𝒳), Covered
products (𝒫), Best solution (𝒳*)
𝒮^LIST ← ∅
𝒳 ← ∅
𝒫 ← ∅
𝒳* ← ∅
procedure RECURSIVE_GREEDY(𝒰, 𝒮,c(𝒮),𝒮^LIST,𝒳,𝒫,𝒳*)
    for i = 1,...,|𝒮^LIST| do
        Pick 𝒮_i^LIST ∈ 𝒮^LIST
        𝒳 ← 𝒳 ∪ {𝒮_i^LIST}
        𝒫 ← 𝒫 ∪ 𝒮_i^LIST
        Update c
        if 𝒫 = 𝒰 then
            Update 𝒳* if current solution is better.
        end if
        Create 𝒮^LIST including points that minimize c(𝒮)/|𝒮−𝒫|
        RECURSIVE_GREEDY(𝒰, 𝒮,c(𝒮),𝒮^LIST,𝒳,𝒫,𝒳*)
    end for
    return 𝒳*
end procedure
```

The second feature of the problem is that the number of subsets are much more than the number of elements. In the original greedy algorithm, one subset is picked in a iteration based on the minimum ratio test. In the product family design, a lot of points are tied for the minimum ratio test. Instead


https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

of randomly choosing a point, recursive greedy algorithm tries every point that have the same ratio. Among the results, the recursive algorithm will return the best solution. In the algorithm, $\mathscr{S}^{LIST}$ is the set of points (subsets) whose minimum ratio are the same. The algorithm will try all the points in the set of candidate points. The set of selected points is denoted as $\mathscr{X}$, and the set of covered products are $\mathscr{P}$. When there is no element in the first set of candidate points, the algorithm will stop and return the best solution, $\mathscr{X}^*$.
The recursive greedy algorithm is similar to the depth-first search for a tree in graph theory. Before the iteration starts, $\mathscr{S}^{LIST}$, which is the set of candidate points is defined as an empty set. It is the root node of the tree. In the first iteration, the first set of candidate points are determined by the minimum ratio test. The elements in the set become children of the root node. The elements in the first set of candidate points are selected as the first subset one by one. Each selected subset creates its own set of candidate points which become children of the selected subset. The algorithm stops when all the nodes in the tree are explored.

## 3 NUMERICAL EXPERIMENT

For numerical experiments, a vehicle design problem of three-degree-of-freedom (3DOF) model is used. The proposed formulation with WSC is solved by the proposed recursive greedy algorithm.

### 3.1 A linear three-degree-of-freedom (3DOF) model

#### *3.1.1 State equations of the linear 3DOF model*

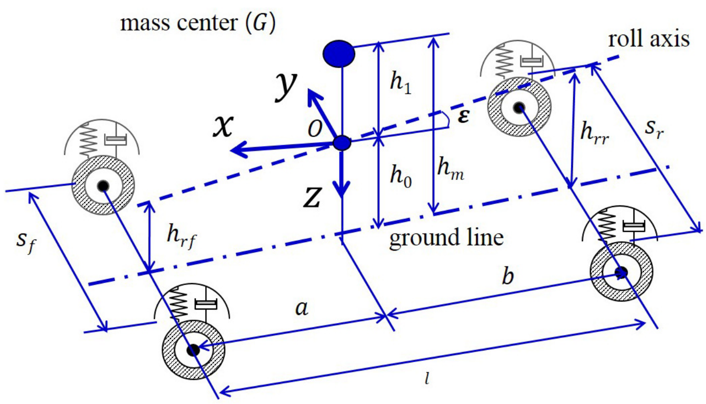


*Figure 1. Geometry diagram of the vehicle model*

A linear 3DOF model is derived for the automotive vehicle design (Peng, 1996). The geometry diagram of the vehicle model is shown in Figure 1. The linear 3DOF model is obtained by adding a roll degree of freedom to a bicycle model. In the model, four state equations are defined with four state variables which are yaw rate ($x_1 = r$), lateral velocity ($x_2 = V$), roll angle ($x_3 = \phi$), and roll rate ($x_4 = p = \dot{\phi}$). The four equations of motion are derived from lateral dynamics ($Y$), roll moment ($N$), yaw moment ($N$), and the roll kinematics. The roll kinematics is simply defined as $\dot{x_3} = x_4$. In lateral dynamics, a linear tire model is used. The total side-force in front axle is

$$F_{yf} = C_{\alpha f}\,\alpha_f + (C_{\gamma f}\,\epsilon_{cf})\phi \tag{3}$$

where $C_{\alpha f}$ is the tire cornering stiffness, $\alpha_f$ is the slip angle, $C_{\gamma f}$ is the tire camber stiffness, and $\epsilon_{cf}$ is the suspension roll-camber coefficient. With the front roll-steer coefficient as $\epsilon_{rf}$, the slip angle can be defined as

$$\alpha_f = \delta + \epsilon_{rf}\,\phi - \frac{V + ar}{U} \tag{4}$$


https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

where $U$ is the longitudinal velocity, $\delta$ is the steering angle, and $a$ is a distance between the front axle and the center of mass. By Equations 3 and 4 and Newton's law of motion, an equation is derived as follows.

$$\begin{aligned} Y &= F_{yf} + F_{yr} \\ &= C_{\alpha f}\left((\delta + \epsilon_{rf}\phi) - \frac{V+ar}{U}\right) + (C_{\gamma f}\epsilon_{cf})\phi + C_{\alpha r}\left((\delta + \epsilon_{rr}\phi) - \frac{V+ar}{U}\right) + (C_{\gamma r}\epsilon_{cr})\phi \\ &\equiv Y_V V + Y_r r + Y_\phi \phi + Y_p p + Y_\delta \delta \\ &= M(\dot{V} + \dot{p}h_1 + rU) \end{aligned} \tag{5}$$

Here, $Y_V$, $Y_r$, $Y_\phi$, $Y_p$, and $Y_\delta$ are coefficients of $V$, $r$, $\phi$, $p$ ,and $\delta$ respectively. Also, $h_1$ is the height of the mass center above the roll axis. In a similar way, the rest two equations can be derived by the total roll moment ($L$), the total yaw moment ($N$). The equations are

$$\begin{aligned} L &= L_{suspension} + L_{weight} \\ &= -(K_{\phi f} + K_{\phi r})\phi - (B_{\phi f} + B_{\phi r})p + Wh_1\phi \quad \equiv L_V V + L_r r + L_\phi \phi + L_p p + L_\delta \delta \\ &= I_{xx}\dot{p} - I_{xz}\dot{r} + Mh_1(\dot{V} + rU) \end{aligned} \tag{6}$$

$$\begin{aligned} N &= aF_{yf} - bF_{yr} \\ &= a\left\{C_{\alpha f}\left((\delta + \epsilon_{rf}\phi) - \frac{V+ar}{U}\right) + (C_{\gamma f}\epsilon_{cf})\phi\right\} - b\left\{C_{\alpha r}\left((\delta + \epsilon_{rr}\phi) - \frac{V+ar}{U}\right) + (C_{\gamma r}\epsilon_{cr})\phi\right\} \\ &\equiv N_V V + N_r r + N_\phi \phi + N_p p + N_\delta \delta \\ &= I_{zz}\dot{r} - I_{xz}\dot{p} \end{aligned} \tag{7}$$

where $b$ is a distance between the rear axle and the center of mass, $K_{\phi f}$ is the front suspension roll stiffness, $B_{\phi f}$ is the front suspension roll damping. By Equations 5 - 7 and the roll kinematics, a matrix format of the state equations is following.

$$\dot{x} = (M_1^{-1}M_2)x + (M_1^{-1}M_3)u \tag{8}$$

$$M_1 = \begin{bmatrix} 0 & M & 0 & Mh_1 \\ -I_{xz} & Mh_1 & 0 & I_{xx} - \epsilon I_{xz} \\ I_{zz} & 0 & 0 & -I_{xz} + \epsilon I_{zz} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad M_2 = \begin{bmatrix} Y_r - MU & Y_V & Y_\phi & Y_P \\ L_r - Mh_1 U & L_V & L_\phi & L_P \\ N_r & N_V & N_\phi & N_P \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad M_3 = \begin{bmatrix} Y_\delta \\ L_\delta \\ N_\delta \\ 0 \end{bmatrix}$$

In the linear 3DOF model, we categorize the parameters into two groups, which are design variables and the design parameters. Design variables are the design factors that are directly related to the component and properties that are used to design components. Design parameters are fixed values for the products, which are defined according to the vehicle types. Table 3 shows the values for design parameters that are used for all types of vehicles in this product family. We set front roll steer, rear roll steer, front roll camber, and rear roll camber 0.27, $-0.03$, $-0.29$, and 0.23 in deg$/1000N$. The values in Table 3 are converted to rad$/N$. For the front camber stiffness and the rear camber stiffness, they are set as $-65$ and $-40$ in $N/$deg for one tire. The values are converted to $N/$deg.

#### *3.1.2 Product family design in the linear 3DOF model*

In this numerical experiment, eight types of cars are designed based on three criteria (sedan vs. SUV, normal vs. sports, light vs. heavy). Design parameters are determined depending on the characteristics of the products. For example, the height of the mass center above ground for sedans is higher than that


https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

for SUVs. Also, the values of wheelbase, total weight at the front and rear axle, and yaw moment of inertia are assigned appropriately as shown in Table 4.

*Table 3. Common design parameters*

| | Description | Value (Unit) | | Description | Value (Unit) |
|---|---|---|---|---|---|
| $I_{xx}$ | Roll inertia about mass center | 200 ($kgm^2$) | $\epsilon_{rf}$ | Front roll steer | 4.71e−6 (rad/$N$) |
| $I_{xz}$ | Product of inertia | 0 | $\epsilon_{rr}$ | Rear roll steer | −5.24e−7 (rad/$N$) |
| $b_f$ | Front suspension damping rate | 1100 ($n/(m/s)$) | $\epsilon_{cf}$ | Front roll camber | −5.06e−6 (rad/$N$) |
| $b_r$ | Rear suspension damping rate | 1200 ($n/(m/s)$) | $\epsilon_{cr}$ | Rear roll camber | 4.01e−6 (rad/$N$) |
| $s_f$ | Front spring and damper spacing | 1.2 ($m$) | $C_{\gamma f}$ | Front camber stiffness | −7448.45 ($N$/rad) |
| $s_r$ | Rear spring and damper spacing | 1.2 ($m$) | $C_{\gamma r}$ | Rear camber stiffness | −4583.66 ($N$/rad) |
| $h_{rf}$ | Height of front roll center | 0.2 ($m$) | $h_{rr}$ | Height of rear roll center | 0.5 ($m$) |

*Table 4. Design parameters for different vehicle types*

| Products | | Sedan | | | | SUV | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Normal | | Sports | | Normal | | Sports | |
| | | Light | Heavy | Light | Heavy | Light | Heavy | Light | Heavy |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $h_m$ | height of mass center above ground ($m$) | 0.56 | 0.56 | 0.56 | 0.56 | 0.62 | 0.62 | 0.62 | 0.62 |
| $l$ | wheelbase ($m$) | 2.4 | 2.4 | 2.4 | 2.4 | 2.7 | 2.7 | 2.7 | 2.7 |
| $w_f$ | total weight at front axle ($kg$) | 725 | 825 | 725 | 825 | 850 | 925 | 850 | 925 |
| $w_r$ | total weight at rear axle ($kg$) | 575 | 625 | 575 | 625 | 700 | 725 | 700 | 725 |
| $I_{zz}$ | Yaw moment of inertia ($kgm^2$) | 2250 | 2350 | 2250 | 2350 | 2450 | 2550 | 2450 | 2550 |

Four performance functions are used to evaluate vehicles. They are the roll gain ($R_1$,deg/$g$), steering sensitivity ($R_2$, $(m/s^2)/100$deg), steady state roll angle ($R_3$,deg), and yaw rate gain ($R_4$, 1/sec). There are six design variables which are directly related to the components in the model; front suspension stiffness ($\xi_1$, $N/m$), rear suspension stiffness ($\xi_2$, $N/m$), front stabilizer bar ($\xi_3$, $Nm$/deg), rear stabilizer bar ($\xi_4$, $Nm$/deg), front cornering stiffness ($\xi_5$, $N$/deg), and rear cornering stiffness ($\xi_6$, $N$/deg). Boundaries of the design variables and target ranges of performance functions are set differently by the characteristics of the vehicles. For sedans, we set higher target ranges for $R_2$ and $R_4$, and lower target ranges for $R_1$ and $R_3$ so that the sedans achieves faster handling response than SUV. In the same way, light vehicles and sports vehicles have higher targets for $R_2$ and $R_4$, and lower targets for $R_1$ and $R_3$ than heavy vehicles and normal vehicles.
The goal of the experiment is maximizing commonality among the products (i.e., minimizing the total number of components). With the WSC formulation and the proposed method, values for each design variable are obtained. Designers can create suspensions, stabilizer bars, and tire components with the obtained values. The target ranges and boundaries for each product are shown in Table 5.

### 3.1.3 Result

In this experiment, eight products are produced with the maximum commonality of components. The optimization problem is solved with the proposed method, recursive greedy algorithm. The result is displayed in Table 6.
Before solving the optimization problem, the design points that satisfy the target ranges of products are obtained. The sampled design point is checked its feasibility for every product. In the design exploration, 8049 points are sampled in total. The points are discretized with grid points. 7781 design points and 268 components are obtained from the discretization.
The result shows that 24 components are required to produce eight types of vehicles. From the table, designers can build components accordingly. For example, to make front tires that are needed to build eight types of vehicles, the values of $\xi_5$ should be examined. Since it is a simple 3DOF model, the front cornering stiffness is the only design parameter that differentiates the attribute of the front tire. In the result, we have three different values for $\xi_5$ (981.25, 1168.75, and 1181.25). Then, the designer will make three different type of tires, and it is sufficient for the front tire to build eight different vehicles.


https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

Table 5. Product specifications

| Products | | | | | Variables | | | | | | Performance functions | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | $\xi_1$ | $\xi_2$ | $\xi_3$ | $\xi_4$ | $\xi_5$ | $\xi_6$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ |
| Sedan | Normal | Light | 1 | LB | 16000 | 17600 | 640 | 480 | 950 | 850 | 2.999 | 0.978 | -1.247 | 49.53 |
| | | | | UB | 24000 | 26400 | 960 | 720 | 1050 | 950 | 3.117 | 1.023 | -1.199 | 50.72 |
| | | Heavy | 2 | LB | 16000 | 17600 | 640 | 480 | 950 | 850 | 3.275 | 0.953 | -1.361 | 46.31 |
| | | | | UB | 24000 | 26400 | 960 | 720 | 1050 | 950 | 3.403 | 0.995 | -1.310 | 47.47 |
| | Sports | Light | 3 | LB | 16000 | 17600 | 765 | 605 | 1150 | 950 | 2.913 | 0.997 | -1.210 | 54.44 |
| | | | | UB | 24000 | 26400 | 1085 | 845 | 1250 | 1050 | 3.024 | 1.040 | -1.165 | 55.68 |
| | | Heavy | 4 | LB | 16000 | 17600 | 765 | 605 | 1150 | 950 | 3.121 | 0.976 | -1.292 | 51.53 |
| | | | | UB | 24000 | 26400 | 1085 | 845 | 1250 | 1050 | 3.230 | 1.020 | -1.248 | 52.64 |
| SUV | Normal | Light | 5 | LB | 20000 | 22400 | 720 | 560 | 950 | 850 | 3.531 | 0.979 | -1.467 | 45.54 |
| | | | | UB | 30000 | 33600 | 1080 | 840 | 1050 | 950 | 3.668 | 1.024 | -1.413 | 46.76 |
| | | Heavy | 6 | LB | 20000 | 22400 | 720 | 560 | 950 | 850 | 3.818 | 0.915 | -1.588 | 42.66 |
| | | | | UB | 30000 | 33600 | 1080 | 840 | 1050 | 950 | 3.970 | 0.957 | -1.527 | 43.80 |
| | Sports | Light | 7 | LB | 20000 | 22400 | 845 | 685 | 1150 | 950 | 3.429 | 0.983 | -1.421 | 50.20 |
| | | | | UB | 30000 | 33600 | 1205 | 965 | 1250 | 1050 | 3.552 | 1.025 | -1.372 | 51.30 |
| | | Heavy | 8 | LB | 20000 | 22400 | 845 | 685 | 1150 | 950 | 3.686 | 0.948 | -1.527 | 47.44 |
| | | | | UB | 30000 | 33600 | 1205 | 965 | 1250 | 1050 | 3.816 | 0.988 | -1.474 | 48.50 |

Table 6. Result of 3DOF problem

| | Product | | | | | | | | Components |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Total |
| $\xi_1$ | 18100 | 18500 | 23700 | 24900 | 23700 | 18100 | 24900 | 24500 | 24 |
| $\xi_2$ | 26300 | 22300 | 26300 | 29900 | 20300 | 21300 | 26300 | 27500 | |
| $\xi_3$ | 831.25 | 831.25 | 943.75 | 831.25 | 831.25 | 943.75 | 981.25 | 981.25 | |
| $\xi_4$ | 506.25 | 581.25 | 706.25 | 706.25 | 706.25 | 706.25 | 893.75 | 893.75 | |
| $\xi_5$ | 981.25 | 981.25 | 981.25 | 981.25 | 1168.75 | 1181.25 | 1181.25 | 1181.25 | |
| $\xi_6$ | 881.25 | 881.25 | 881.25 | 881.25 | 968.75 | 993.75 | 993.75 | 993.75 | |

## 4 DISCUSSION

In the proposed method, we assumed that one attribute defines the characteristics of a component. In the numerical experiment, the rear tire is manufactured based on the values of the rear cornering stiffness. In the 3DOF vehicle model, there is another factor that determines the characteristics of the tire, which is the rear camber stiffness. As we set a constant value for the parameter, we can assume that the tires that have the same rear cornering stiffness are the identical tires. However, if the rear camber stiffness becomes a design variable, the cost function of the proposed method should be modified. Let's say there is another variable $\xi_7$ that is the rear camber stiffness. In this case, the pair of the variables, both $\xi_6$ and $\xi_7$ will determine the attribute of the rear tire. The updating scheme of the cost function will be different for this case.

In the preprocessing, we assumed continuous attributes for design variables. If attributes of the components are discrete or have countable options, designers can obtain the candidate designs by trying the different combinations of the attributes. For example, designers would like to choose existing tires that are already available in practice instead of manufacturing there own tire. In this case, they can create candidate designs with existing tires. If the number of options is too huge to create all the combinations of attributes, they can assume the attribute as continuous variables and choose the existing tire that has the most similar attribute to the optimal solution.

In product family design, modularity can be considered to build better platforms. Since there is a trade-off between modularity and commonality, a multi-objective problem can be solved to achieve Pareto optimality. Rather than considering modularity as an objective, it can be considered as constraints as if target ranges of performance functions are constraints in the proposed method. From the design structure matrix (DSM), designers can have optimal modularity for each product. If the number of components that violate the optimal modularity is evaluated, the constraint for the modularity can be added to the


https://doi.org/10.1017/dsi.2019.302 Published online by Cambridge University Press

integer problem in section 3. The violation can be calculated by building a matrix that represents relationships between products and components. For each component, there are restrictions on sharing the component among the product based on the optimal modularity.
An objective of the proposed method is to maximize the commonality of the product family. It is done by setting the cost function as the number of components. The cost function can be set in different ways, such as considering manufacturing cost of components or economics of scale. The manufacturing cost of components can be easily adapted to the proposed method if designers have the price information of each component. For the economics of scale, we have to analyze the demand of the markets. To maximize profit, demand model should be studied to meet needs in various markets. By analyzing market data, the market segments can be obtained by clustering so that designers can determine the product positioning (Lei and Moon, 2015).

## 5 CONCLUSION

We utilize WSC to formulate product family design. The greedy algorithm for WSC is modified because of the generalized cost and the property of the formulation. Targets for the performance functions are set as constraints of designing products, while the commonality is maximized. The proposed method can be applied directly if designers already have candidates of the components. Otherwise, designers can sample design points that satisfy the target ranges of performance functions, and discretize the problem to apply the proposed method as it is done in section 2.2. With the proposed algorithm, the optimal attributes of the components that minimize the number of distinct components are obtained.

## REFERENCES

Simpson, T.W., 2004. "Product platform design and customization: Status and promise." *Ai Edam*, Vol. 18 No. 1, pp. 3–20.
Simpson, T.W., Maier, J.R. and Mistree, F., 2001. "Product platform design: method and application." *Research in engineering Design*, Vol. 13 No. 1, pp. 2–22.
Du, X., Jiao, J. and Tseng, M.M., 2001. "Architecture of product family: fundamentals and methodology." *Concurrent Engineering*, Vol. 9 No. 4, pp. 309–325.
Gonzalez-Zugasti, J.P., Otto, K.N. and Baker, J.D., 2000. "A method for architecting product platforms." *Research in engineering design*, Vol. 12 No. 2, pp. 61–72.
Fujita, K. and Yoshida, H., 2004. "Product variety optimization simultaneously designing module combination and module attributes." *Concurrent Engineering*, Vol. 12 No. 2, pp. 105–118.
Baylis, K., Zhang, G. and McAdams, D.A., 2018. "Product family platform selection using a Pareto front of maximum commonality and strategic modularity." *Research in Engineering Design*, pp. 1–17.
Rojas Arciniegas, A.J. and Kim, H.M., 2011. "Optimal component sharing in a product family by simultaneous consideration of minimum description length and impact metric." *Engineering Optimization*, Vol. 43 No. 2, pp. 175–192.
Eichstetter, M., Müller, S. and Zimmermann, M., 2015. "Product family design with solution spaces." *Journal of Mechanical Design*, Vol. 137 No. 12, p. 121401.
Chvatal, V., 1979. "A greedy heuristic for the set-covering problem." *Mathematics of operations research*, Vol. 4 No. 3, pp. 233–235.
Peng, H., 1996. *Lecture Notes for ME542: Vehicle Dynamics.* Mechanical Engineering Department, University of Michigan.
Lei, N. and Moon, S.K., 2015. "A Decision Support System for market-driven product positioning and design." *Decision Support Systems*, Vol. 69, pp. 82–91.
Rai, R. and Allada, V., 2003. "Modular product family design: agent-based Pareto-optimization and quality loss function-based post-optimal analysis." *International Journal of Production Research*, Vol. 41 No. 17, pp. 4075–4098.
Simpson, T.W. and D'souza, B.S., 2004. "Assessing variable levels of platform commonality within a product family using a multiobjective genetic algorithm." *Concurrent Engineering*, Vol. 12 No. 2, pp. 119–129.
Khajavirad, A., Michalek, J.J. and Simpson, T.W., 2009. "An efficient decomposed multiobjective genetic algorithm for solving the joint product platform selection and product family design problem with generalized commonality." *Structural and Multidisciplinary Optimization*, Vol. 39 No. 2, pp. 187–201.
Chowdhury, S., Messac, A. and Khire, R.A., 2011. "Comprehensive product platform planning (cp3) framework." *Journal of Mechanical Design*, Vol. 133 No. 10, p. 101004.
Nelson, S.A., Parkinson, M.B. and Papalambros, P.Y., 2001. "Multicriteria optimization in product platform design." *Journal of Mechanical Design*, Vol. 123 No. 2, pp. 199–204.