# A Decomposed Genetic Algorithm for Solving the Joint Product Family Optimization Problem

Aida Khajavirad[*] and Jeremy J. Michalek[†]
*Carnegie Mellon University, Pittsburgh, PA 15213 USA*

Timothy W. Simpson[‡]
*The Pennsylvania State University, University Park, PA, 16802 USA*

**A critical step when designing a successful product family is to determine a cost-saving platform configuration along with an optimally distinct set of product variants that target different market segments. Numerous optimization-based approaches have been proposed to help resolve the tradeoff between platform commonality and the ability to achieve distinct performance targets for each variant. However, the high dimensionality of an "all-in-one" algorithm for optimizing the joint problem of 1) platform variable selection, 2) platform design and 3) variant design makes most of these approaches impractical when a large number of products is considered. Many existing approaches have restricted the scope of the problem by fixing platform configuration *a priori*, limiting platform configuration to an all-or-none component sharing strategy, or by solving subsets of the joint problem in stages, sacrificing optimality. In this study, we propose a single-stage optimization approach for solving the joint product family problem with generalized commonality using an efficient decomposition solution strategy involving multi-objective genetic algorithms (MOGAs). The proposed approach overcomes prior limitations by introducing a generalized two-dimensional commonality chromosome and decomposing the joint formulation into a two-level GA, where the upper-level determines the optimal platform configuration while each lower-level designs one of the individual variants in the family. Moreover, all sub-problems run in parallel, and the upper-level GA coordinates consistency among the lower-levels using the MPI (Message Passing Interface) library. The proposed approach is demonstrated by optimizing a family of three general aviation aircraft, and results outperform those from a non-decomposed GA. Results also show that the commonality-performance Pareto front contains solutions with generalized commonality, suggesting the need to avoid all-or-none component sharing restrictions in order to avoid sub-optimality. Future work in scaling the decomposed GA to larger product families is also discussed.**

## I.    Introduction

MARKETPLACE globalization, the proliferation of niche markets driven by heterogeneity of preferences, increased competitive pressures, and demand for differentiated and customized products have rendered the practice of isolated design and production of individual products nearly obsolete. Across many industries, the prevailing practice is to design lines or families of product variants that exploit commonality to take advantage of economies of scale and scope while targeting a variety of market segments and achieving strategic market coverage to deter competitors. Planning families of products requires particular care and attention, since each product competes for market share not only with competitor products, but also with other products in the family.

Generally speaking, a *product family* is a group of related products (i.e., variants) that are derived from a common set of components, modules, and/or subsystems to satisfy a variety of market niches, and the key to a successful product family is the *product platform* around which the product family is derived[1]. Designing a product

---

[*] Graduate Research Assistant, Mechanical Engineering, Student Member AIAA, Email: aida@cmu.edu
[†] Assistant Professor, Mechanical Engineering, Member AIAA, Email: jmichalek@cmu.edu.
[‡] Professor of Mechanical Engineering and Industrial Engineering, Senior Member AIAA, Email: tws8@psu.edu.

platform and corresponding family of products is a difficult task that embodies all of the challenges of product design while adding the complexity of coordinating the design of multiple products in an effort to increase commonality across the set of products without compromising their individual performance. This challenge manifests early in the design process wherein designers must not only specify the *platform configuration* (i.e., selecting which design variables are shared across the products in the family – also referred to as *platform variable selection* or *platform selection*[2]), but also optimize the design of the platform and the individual variants by choosing design variable values while maintaining commonality defined in the platform configuration. Resolving the inherent tradeoff between platform commonality and product distinctiveness is paramount: Increasing the degree of commonality among variants in a product family generally reduces total cost, but it can also compromise the ability of each variant to fully achieve the desired characteristics that make it distinct and attractive to different market segments.

In the next section, we review related work in design optimization that has been developed to map out this tradeoff between commonality and distinctiveness. In Section III we propose an "all-in-one" MOGA for solving the joint product family problem with a generalized commonality chromosome that allows commonality among subsets of the variants. In Section IV we decompose the GA to dramatically improve search efficiency and scalability by reducing the search space of each sub-GA and enabling use of parallel processing. In Section V, an example involving the design of a family of general aviation aircraft is presented and optimized using both sequential and parallel algorithms for comparison. Closing remarks and future work are given in the final section.

## II.    Review of Related Literature

Numerous optimization approaches have been developed within the engineering design community during the past decade to help solve the product family design problem. Simpson[3] reviews and classifies 40 approaches from the literature. In many of these approaches, the design variables that define the product platform within the family are known or specified *a priori*, i.e., before performing the optimization (Allada and Jiang[4], Blackenfelt[5], Chang and Ward[6], D'souza and Simpson[7], Dai and Scott[8], Farrell and Simpson[9], Fellini *et al.*[10], Fujita *et al.*[11], Gonzales-Zugasti *et al.*[12,13], Hernandez *et al.*[14], Kokkolaras *et al.*[15], Kumar *et al.*[16], Li and Azarm[17], Messac *et al.*[18], Nelson *et al.*[19], Ortega *et al.*[20], Seepersad *et al.*[21,22], Simpson *et al.*[23,24]), whereas in other instances, the platform variable selection is determined during optimization, i.e., the platform is specified *a posteriori* (Akundi *et al.*[25], Cetin and Saitou[26], Dai and Scott[8], de Weck *et al.*[27], Fellini *et al.*[28,29], Fujita and Yoshida[30], Fujita *et al.*[31], Gonzales-Zugasti and Otto[32], Hernandez *et al.*[33,34], Messac *et al.*[35], Nayak *et al.*[36], Rai and Allada[37], Hassan *et al.*[38], Simpson and D'souza[7], Khire *et al.*[2]). In a similar manner, Fujita[39] has divided product variety optimization problems into three classes: In Class-I problems, product attributes are optimized under a fixed platform configuration (i.e., the platform is known *a priori*); Class-II problems deal with finding the optimal module selection using predefined modules (i.e., the design of each module is known *a priori*); and finally, in Class-III problems, the product attributes and platform configuration are optimized simultaneously. We refer to this Class III, *a posteriori* problem as the *joint product family optimization problem* because it involves determining the optimal combination of 1) platform variable selection, 2) platform design and 3) variant design. Since each of these decisions is generally dependent on the others, approaches with restricted scope cannot guarantee optimality, except in special cases, and there is a clear need for an algorithm capable of solving the joint problem for practical product family applications.

Simpson[3] classifies optimization-based approaches for solving the product family design problem where platform variable selection is specified *a posteriori* based on the number of stages involved: *single-stage approaches* optimize the product platform and resulting family of products simultaneously (Akundi *et al.*[25], Cetin and Saitou[26], Fujita *et al.*[31], Fujita and Yoshida[30], Gonzales-Zugasti and Otto[32], Hassan *et al.*[38], Simpson D'souza[7], Khire *et al.*[2]) while *two-stage approaches* optimize the platform variables first and then instantiate the individual variants based on this platform during the second stage of the optimization (Dai and Scott[8], De Weck *et al.*[27], Hernandez *et al.*[33,34], Messac *et al.*[35], Nayak *et al.*[36], Rai and Allada[37]). A special case is Fellini *et al.*[28,29], who solve the joint problem by specifying platform variable selection in the first stage and optimizing the platform and variant variables in the second stage. While both single- and two-stage approaches can be effective at determining design variable settings

for the product family, two-stage approaches may lead to sub-optimal solutions in general, therefore single-stage approaches are preferred on the criterion of optimality. A major disadvantage of single-stage approaches, however, is that the dimensionality of the optimization problem is considerably higher compared to two-stage approaches, which makes these approaches impractical when designing a family with a large number of products. In summary, an efficient approach is needed for solving the joint product family optimization problem in a single-stage while effectively handling problems involving large numbers of products.

Among prior product family optimization approaches, many studies have found that Genetic Algorithms (GAs) are well-suited for optimizing a product platform and its corresponding family of products. Li and Azarm[17] were among the first in the engineering design community to use GAs to solve the product family design problem, but they required specification of the platform *a priori*. D'Souza and Simpson[40] developed a similar approach using the Non-Dominated Sorting GA (NSGA-II)[41], which was then extended using an augmented chromosome string to examine varying levels of platform commonality to simultaneous determine platform variable selection and design of the platform and variants.[7] In particular, each chromosome string was augmented by $n$ additional commonality controlling genes, where $n$ is the number of candidate platform variables. Each of these commonality variables can take the value of either 0 or 1, where a value of 1 means that the corresponding design variable is made common among all of the products in the family, while a value of 0 allows design variables to be unique among products. A commonality metric was defined by summation of design variables' variations within the product family. According to this definition, the commonality in the resulting product family is based not only on how many variables are common but also on how similar the values of unique variables are to one another. Total deviation from each product performance target and the commonality metric were treated as separate objective functions. This approach was then applied to the design of a family of three general aviation aircraft, as defined by their seating capacity. Results successfully showed the tradeoff between commonality and performance in the product family. Finally, Hassan *et al.*[38] used the *N-Branch Tournament Selection GA* for optimizing a family of three commercial satellites. They imposed no explicit representation for controlling commonality within the family, and the chromosome included only three sub-strings, each one representing the design variables for a specific satellite in the product line. The commonality metric was evaluated based on the number of common design variables shared by all three sub-strings. The problem had four objectives: the launch mass of each satellite in the product line and the commonality metric. However, all these prior approaches have two primary limitations that restrict their scope of applicability: 1) They restrict commonality decisions to a single platform, and 2) they have limited scalability for solving large problems.

The first restriction, which limits commonality decisions for each variable to be either a "full" platform variable shared by all variants or a non-shared variable that is different among all variants, is useful for simplicity but causes unnecessary restriction in practice: There are many cases where allowing commonality among a subset of variants may lead to more efficient design opportunities. To overcome this limitation, we generalize Simpson and D'Souza's[7] commonality controlling gene by introducing a two-dimensional chromosome that accounts for commonality of each module among any subset of the variants.

The second limitation of the aforementioned approaches is limited scalability to large problems with many product variants. As the number of variants increases, the chromosome string becomes long, and solving a high-dimensional space with a single GA can be difficult, if not impossible. For instance, Akundi and Simpson[25] found that as many as 25,000 generations – with population sizes of 1,500 – were needed to obtain a good spread of solutions for a family of 10 universal motors that had three conflicting objectives, where each motor was defined by 8 design variables. The addition of generalized commonality in our formulation also adds significant complexity, making an all-in-one approach intractable in many cases. Therefore, in order to create a general and scalable algorithm that can be applied to a large number of products without losing performance, we propose a method for decomposing the initial "all-in-one" GA into an upper-level GA that controls commonality decisions and a set of lower-level GAs: one for the design variables of each product. Each lower-level GA deals only with an individual product and its chromosomes consist only of the design variables for that product. Commonality constraints are imposed from the upper-level GA to the lower-level ones to enforce commonality decisions on the values of design variables across products. Moreover, the decomposed model can be parallelized by executing each GA on a separate

American Institute of Aeronautics and Astronautics

processor and using the MPI (Message Passing Interface) library for exchanging data among sub-problems. Hence, in addition to improved performance due to using decomposition, it is possible to achieve a dramatic reduction in computational time using parallel processing. In the next section we describe the "all-in-one" MOGA with the generalized commonality chromosome, and we decompose the GA in Section IV.

## III.    Proposed Approach

In this paper, we introduce a MOGA formulation for determining the Pareto front representing the tradeoff between commonality and individual variant performance in the family. The underlying algorithm for our MOGA code is the elitist non-dominated sorting GA (NSGA-II) introduced by Deb[41], which has been shown to be capable of finding a well-converged and well-distributed set of Pareto optimal solutions in a reasonable computational time for many problems. In this algorithm, the offspring population $Q$ of $N$ individuals is first created from the parent population $P$ using roulette-wheel selection, simulated binary cross-over and polynomial mutation operators. Then $P$ and $Q$ are combined together to form a population $R$ composed of $2N$ individuals ($R=P\cup Q$). A non-dominated sorting procedure is used to classify and sort the individuals in $R$ into successive non-dominated fronts. The first front is the set of individuals for which no other individual in the population has a higher fitness value for all objectives; the second front is the non-dominated set of the remaining designs that are not in the first front; and so on. Finally, individuals are copied into a new population $S$ of size $N$ starting with the first non-dominated front, continuing with the second non-dominated front, and so on. Because $R$ is twice as big as $S$, $S$ will not include all design points. When the last allowed front is considered, there may exist more solutions in the last front than remaining slots in the new population. Instead of arbitrarily discarding some members from the last front, the points that reside in the least-crowded region in that front (having a larger crowding distance) are chosen to encourage diversity. Using the concepts of non-dominated sorting and crowding distance in both selection and replacement schemes makes this algorithm efficient in finding a well-distributed Pareto optimal front. However, in order to apply the original NSGA-II code to the product family problem we have modified the chromosome representation, crossover, and mutation operators as described in the sections that follow.

### A.  Chromosome Representation

As mentioned in Section II, we generalize the commonality controlling gene approach of Simpson and D'Souza[7] to relax the all-or-none component sharing restriction so that platform variables can be shared among any subset of product variants. This generalization was achieved by introducing two parallel chromosomes for each individual in the MOGA population (see ). In this representation, the first chromosome, which we call the "commonality chromosome," is a two-dimensional chromosome that defines the platform configuration. The second chromosome contains design variables of all variants in the family and defines the optimal values of each individual product design variables. The algorithm ensures that the two chromosomes remain consistent during evolution using constraints imposed by the commonality chromosome on the design variable chromosome. Hence, in a product family with $p$ products, each defined by $n$ design variables, the commonality chromosome is a two dimensional matrix with $p$ rows and $n$ columns while the design variable chromosome contains $np$ genes. The commonality chromosome is generated so that genes can take any integer value between 1 and $p$, where any two equal integer values for the same variable indicate that the corresponding design variables are common. An example of this representation for a product family with three products and six design variables is shown in .

|  | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|---|---|---|---|---|---|---|
| **Product 1** | 1 | 2 | 3 | 1 | 2 | 1 |
| **Product 2** | 2 | 2 | 3 | 2 | 3 | 3 |
| **Product 3** | 3 | 1 | 3 | 1 | 1 | 3 |

$x_1$ is distinct in each product.
$x_2$ is shared between 1st and 2nd products.
$x_3$ is shared among all products.
$x_4$ is shared between 1st and 3rd products.
$x_5$ is distinct in each product.
$x_6$ is shared between 2nd and 3rd products.

**(a) Commonality chromosome**

| $x_{11}$ | $c_2$ | $c_3$ | $c_4$ | $x_{51}$ | $x_{61}$ | $x_{12}$ | $c_2$ | $c_3$ | $x_{42}$ | $x_{52}$ | $c_6$ | $x_{13}$ | $x_{23}$ | $c_3$ | $c_4$ | $x_{53}$ | $c_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| **Design Variable Values for Product 1** | **Design Variable Values for Product 2** | **Design Variable Values for Product 3** |

**(b) Design variable chromosome**

**Figure 1. Two Parallel Chromosomes for Each Product Family in the GA Population**

## B. Crossover Operators

Due to the 2-D configuration of the commonality chromosome, a *two-dimensional binary crossover operator* was developed, which is a direct extension of the one-point crossover operator to two dimensions. In this operator, two random integer numbers are generated in the range of $(1, p)$ and $(1, n)$ to select crossover sites along $p$ and $n$, where $p$ and $n$ again represent the number of products and number of design variables in each product, respectively. These two random numbers are used to divide the commonality chromosome into four quadrants. Then a third random integer number, in the range of $[1, 4]$, is generated to decide which quadrant is to be interchanged (see ).
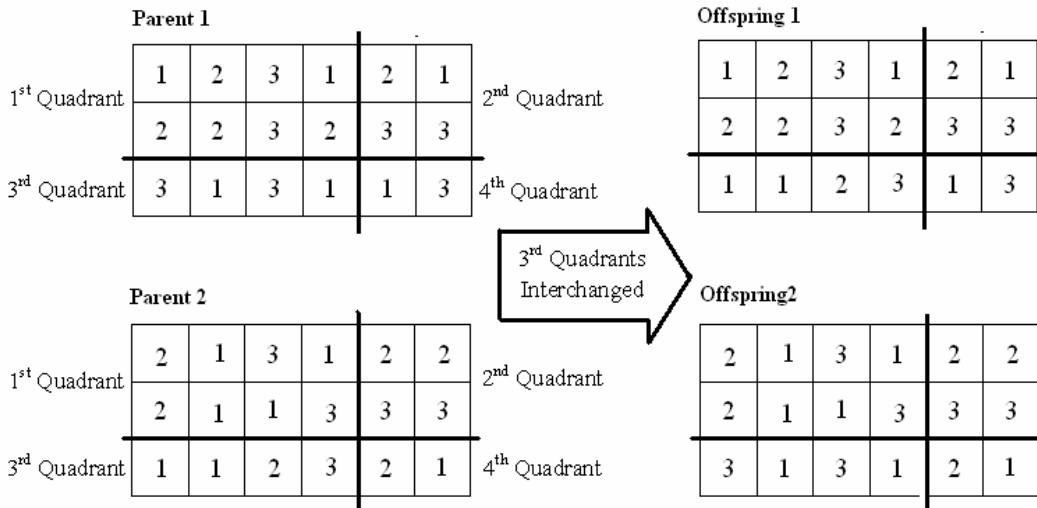


**Figure 2. Two-Dimensional Binary Crossover Operator**

The crossover type applied to the design variable chromosome is the default used in the original NSGA-II code, which is *simulated binary crossover.* After applying crossover to both chromosomes, the algorithm modifies the design variable chromosome based on the constraints that the commonality chromosome imposes on it. In our implementation, we implement this consistency imposition by copying the corresponding value of one of the design

variables onto the others; however, it is possible to use alternative approaches, such as averaging the values across variables and assigning the average value to all common variables.

## C. Mutation Operators

The mutation operator used in this algorithm is designed to mutate the platforms in the product family in order to increase the searching quality of the GA code for exploring various levels and configurations of commonality. First, for each design variable, a random number between 0 and 1 is generated. If its value is less than the user-specified mutation probability, then the corresponding design variable in the product family is mutated. In mutation, a new random number between 0 and 1 is generated. If its value is less than 0.5, then the corresponding design variable in the product family is set as distinct in each product, and the algorithm mutates that design variable for each product in the design variable chromosome according to the *polynomial mutation operator* and modifies the commonality chromosome accordingly. Otherwise, the design variable is made common among all products: A random number is generated and passed to the corresponding genes in the design variable chromosome, and the commonality chromosome is modified as well.

## D. Commonality Objective Function

In order to have the MOGA find the optimal platform configuration, an objective function for measuring the commonality for each family of products is added to the set of performance objective functions. Several metrics for measuring the commonality degree in product families have been proposed reflecting various commonality benefits based on company's focus and standpoint. Khajivarad and Michalek[42] argue that the commonality index (*CI*), introduced by Martin and Ishii[43], captures the cost benefits of commonality better than prior metrics used in product family optimization, and we adopt it here as the commonality objective function. *CI* ranges between 0 and 1 and is a measure of unique parts; that is, a higher value indicates the whole product family was made with a fewer number of unique parts: For a product family with $p$ products each with $n$ components *CI* can be found as follows:

$$CI = 1 - \frac{u-n}{n(p-1)} \tag{1}$$

where $u$ represents the total number of distinct components in the product family. By defining $N_i$ as the number of distinct integers for the $i^{th}$ design variable in the commonality chromosome, Eq. (1) can be reformulated as follow:

$$CI = 1 - \frac{\sum_{i=1}^{n}(N_i - 1)}{n(p-1)} \tag{2}$$

Using the above definition, calculation of the commonality objective function uses only the commonality chromosome while the product performance-related objectives use only the design variable chromosome; this is a key feature that enables decomposition of the proposed GA, as discussed next.

## IV.    Decomposition and Parallelization of the Multi-Objective GA

Aforementioned modifications to the original NSGA-II code make it convenient for optimizing the product family problem with generalized commonality; however, this algorithm is still only practical for problems with a relatively small number of design variables and variants. The commonality generalization also increases this complexity, making the algorithm inefficient in dealing with high-dimensional problems. To address this scalability limitation, we propose a decomposition of the original "all-in-one" formulation (). The method involves allocating the commonality chromosome to an upper-level GA and decomposing the design variable chromosome into its product variants, where each variant is allocated to one of the lower-level sub-GAs. Commonality constraints are imposed from the upper-level GA to lower-level ones. Using this formulation, the dimensionality of each lower-level GA remains constant as more variants are added to the product family. The core advantage of this decomposition is that selection of individuals in the population for producing offspring is made with respect to the

fitness of a *subset* of the full product family, rather than the entire product family chromosome. This property can improve performance dramatically because, for example, a product family with some high performing variants need not have a high fitness over the entire product family in order to pass on information from its high performing variants to the next generation. In the "all-in-one" GA, selection of product families from the population is made with respect to the fitness value of those families; in contrast, the decomposed GA involves 1) selection of sub-chromosomes based on their sub-fitness values for producing offspring and 2) coordination of the sub-GAs after each generation to select the subset of product families from the joint parent-offspring population that will advance to the next generation. Some information on the performance of the overall family is also included in the fitness calculation of each sub-GA in order to avoid divergence in offspring generation, but because each sub-GA does not select on the basis of the fitness of the entire product family, but rather on the basis of a sub-fitness value, each sub-GA can carry over features of high-performing subsets of the full product family chromosome to the next generation. Moreover, due to the parallel nature of this decomposed method, each sub-GA can be executed on a separate processor using the MPI (Message Passing Interface) library for sending and receiving data among nodes after each generation. Since communication cost is negligible compared to computational cost, and performance is improved due to decomposition, we obtain a high speedup in computational time through parallel processing.
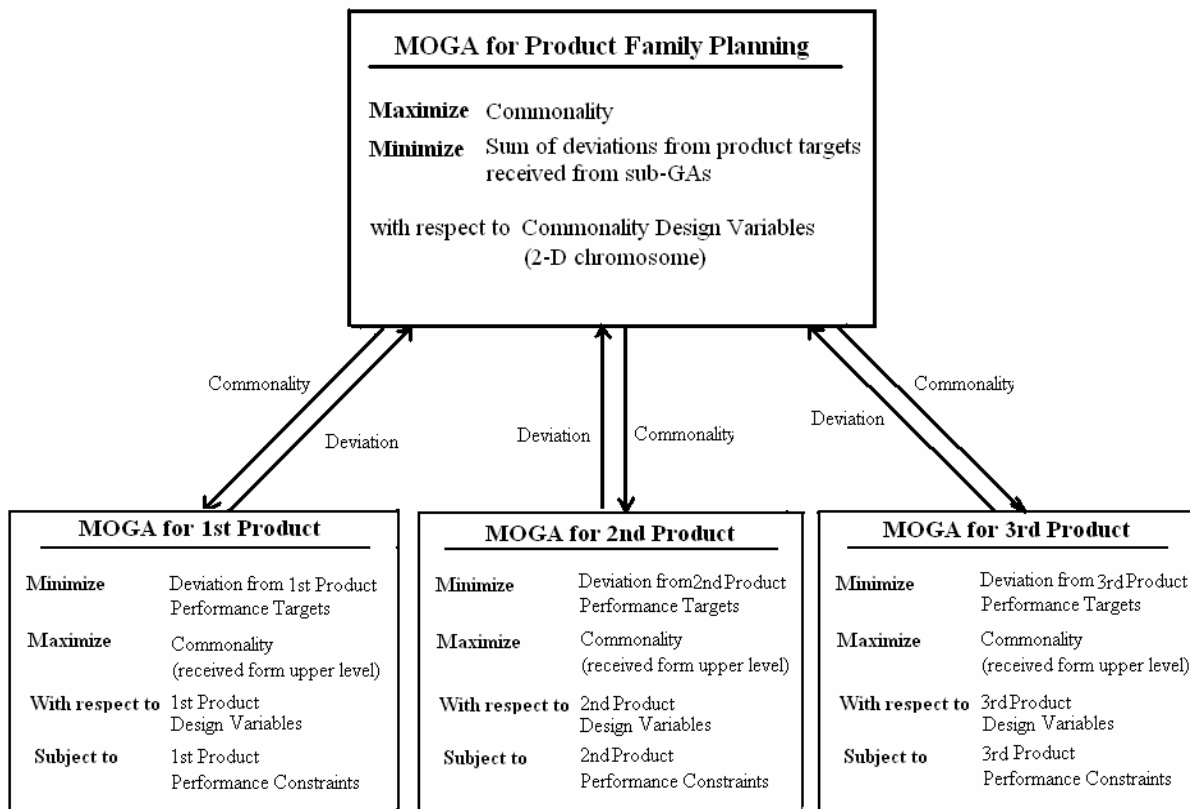


**Figure 3. Decomposed Multi-Objective GA Model for Product Family Design**

The general structure of the proposed model is shown in Figure 3**Error! Reference source not found.**. This figure shows the case of three products; however, the approach can easily be generalized to any number of products. The steps of the algorithm proceed as follows:

**Step 1.** Initial populations are created in the upper-level and lower-level GAs independently. According to the commonality chromosomes, the upper-level GA sends the required data to each lower-level GA, which

American Institute of Aeronautics and Astronautics

modifies its design variable chromosomes to maintain consistency with the corresponding commonality chromosomes.

**Step 2.** The commonality metric, Eq. (2), and individual variant performance objectives are calculated in the upper-level and lower-level GAs, respectively. The upper-level GA sends the commonality metric to all lower-level GAs, which are included in the fitness function of each in addition to the product performance objectives. Each lower-level GA also returns performance deviations to the upper-level GA, and these are summed across variants to form the overall performance objective functions. The passing of these "sub-fitness" values serves to augment fitness values in the individual GA with additional information about the performance of the corresponding family to improve selection for producing offspring.

**Step 3.** Using the selection, crossover and mutation operators, offspring populations are generated in all GAs. The fitness function used for selection in each lower-level GA involves two objectives: the performance of the corresponding variant and the commonality metric for the variant's family. The fitness function used for selection in the upper-level GA involves the fitness of the entire family (i.e.: the commonality metric and the sum of performance deviations for all variants). Crossover and mutation operators at the product level are the same as the sequential version except that the tasks are divided among different processors. For example, the upper-level GA applies the two-dimensional binary crossover operator to the commonality chromosome while lower-level GAs use simulated binary crossover restricted by commonality constraints received from the upper-level. In case of mutation, the upper-level GA decides which design variables should me mutated, i.e., which variables should become common or distinct among the products, and passes this data to the lower-level GAs so that they can mutate the individuals accordingly. The upper-level GA sends the modification data to all lower-level GAs to make all populations consistent, and each lower-level GA passes back the fitness value for its performance objective.

**Step 4.** The upper-level GA combines the parent and offspring population and applies non-dominated sorting to select the best half as the new generation that will define the new population in all lower-level GAs as well.

**Step 5.** If the generation number is equal to the maximum generation number, the algorithm is terminated; otherwise, the process is repeated from Step 2.

## V.  General Aviation Aircraft Example

To demonstrate the proposed approach, we design a family of three general aviation aircraft (GAA), an example that was first introduced by Simpson, *et al.*[44] and later used by Simpson and D'Souza[7] in their aforementioned GA implementations. In this example, a family of three aircraft accommodating 2, 4, and 6 people is optimized. For the purpose of this example, a GAA is defined as a fixed wing, single-engine, single pilot propeller-driven aircraft for 2, 4 and 6 passengers that can cruise at 150 to 300 knots and has a range of 800 to 1000 miles. The design challenge is to determine the best values of the key variables for the fuselage, wing, and engine to satisfy a variety of performance and economic requirements. The performance parameters for an aircraft with a particular set of input variables are obtained from the General Aviation Synthesis Program (GASP) output. GASP, developed by NASA in 1978, is a synthesis and analysis program that facilitates parametric studies of small aircraft[43]. It is specially suited to analyze and study the performance characteristics of small fixed-wing aircraft having a single piston engine, fixed pitch propeller, twin turboprop/turbofan powered business or transport type aircraft. GASP uses an input file configured by the user and creates an output file listing various parameters for a specific aircraft produced through computations and interactions between the sub-modules.

### A. GAA Problem Formulation

Table 1 summarizes the design variables and their respective bounds used in this study. Performance and economic targets and constraints for each aircraft are listed in Tables 2 and 3, respectively. Denoting each product target by $T_j$ and the corresponding GASP response values by $R_j$ and assuming the performance of all products are equally important for us, the goal is to minimize the sum of deviations of responses from target values for the family of products. Hence, for the non-decomposed GA, we can formulate the optimization problem with two objective functions as shown in Eq. (3).

**Table 1. Design Variables and Bounds for GAA Problem**

| Variable Description | Lower Bound | Upper Bound |
|---|---|---|
| **WGS :** Wing Loading, (lb/ft2) | 20.0 | 25.0 |
| **EMCRU :** Design Cruise Speed, (mach) | 0.20 | 0.45 |
| **WS :** Seat Width, (in) | 15.0 | 20.0 |
| **WAS :** Aisle Width, (in) | 17.0 | 20.0 |
| **AR :** Aspect Ratio | 7.0 | 10.0 |
| **SAH :** Horizontal Tail Location on Vertical Tail | 0 | 1 |
| **TCR :** Wing Root Thickness to Chord Ratio | 0.10 | 0.20 |
| **TCT :** Wing Tip Thickness to Chord Ratio | 0.10 | 0.20 |
| **TCHT :** Horizontal Tail Root Thickness to Chord Ratio | 0.08 | 0.15 |
| **ELODT :** Length to Diameter Ratio of Tail Cone of Fuselage | 3 | 4 |
| **DPROP :** Propeller Diameter, (ft) | 5 | 7 |
| **AF :** Activity Factor per Blade | 80.0 | 100.0 |

$$f_1 = 1 - \frac{1}{p} \sum_{i=1}^{p} \left( \frac{1}{n_T} \sum_{j=1}^{n_T} \left| T_{ij} - R_{ij} \right| \right)$$

$$f_2 = 1 - \frac{\sum_{i=1}^{n} (N_i - 1)}{n(p-1)}$$

$n$ **:** Number of design variables (in GAA Example : 12)  (3)

$p$ **:** Number of products (in GAA Example : 3)

$n_T$ **:** Number of Targets (in GAA exapmle : 7)

$T_{ij}$ **:** $j^{th}$ Target of the $i^{th}$ Product

$R_{ij}$ **:** $j^{th}$ Response of the $i^{th}$ Product

$N_i$ : number of variants

**Table 2. Targets for the GAA Problem**

| Targets | 2-Seat | 4-Seat | 6-Seat |
|---|---|---|---|
| 1. Aircraft Fuel Weight (*lbs*) | 450.0 | 400.0 | 350.0 |
| 2. Aircraft Empty Weight (*lbs*) | 1900.0 | 1950.0 | 2000.0 |
| 3. Direct Operating Cost ($*/hr*) | 60.0 | 60.0 | 60.0 |
| 4. Purchase Price ($ in 1970) | 41,000 | 42,000 | 43,000 |
| 5. Lift to Drag Ratio | 17.0 | 17.0 | 17.0 |
| 6. Cruise Speed (*kts*) | 200.0 | 200.0 | 200.0 |
| 7. Range (*nm*) | 2500.0 | 2500.0 | 2500.0 |

**Table 3. Constraints for the GAA Problem**

| Constraints | 2-Seat | 4-Seat | 6-Seat |
|---|---|---|---|
| 1. Maximum Take-off Noise (*db*) | 75.0 | 75.0 | 75.0 |
| 2. Maximum Direct Operating Cost ($*/hr*) | 80.0 | 80.0 | 80.0 |
| 3. Maximum Ride Roughness Coefficient | 2.0 | 2.0 | 2.0 |
| 4. Maximum Aircraft Empty Weight (*lbs*) | 2200.0 | 2200.0 | 2200.0 |
| 5. Maximum Aircraft Fuel Weight (*lbs*) | 450.0 | 475.0 | 500.0 |
| 6. Minimum Flight Range (*nm*) | 2000.0 | 2000.0 | 2000.0 |

Constraints are handled using an exterior penalty method, which in case of multi-objective optimization should be added to all of the objective functions (in our case $f_1$ and $f_2$). Therefore, the new objective functions are formulated as follows:

$$\begin{cases} \phi_1 = f_1 + igen \times \sum_{j=1}^{m} <g_j>^2 \\ \phi_2 = f_2 + igen \times \sum_{j=1}^{m} <g_j>^2 \end{cases}, \quad <g_j> = \begin{cases} g_j & g_j > 0 \\ 0 & g_j \leq 0 \end{cases} \tag{4}$$

where *igen* and $g_j$ represent the generation number and the $j^{th}$ inequality constraint respectively. In the case of the parallel multi-objective GA, in the lower-level GAs, the first objective function for each product is the sum of the normalized deviations from the product's targets:

$$f_1 = 1.0 - \frac{1}{n_T} \sum_{i=1}^{n_T} \left| 1 - \frac{T_i}{R_i} \right| \tag{5}$$

The second objective function is the commonality measurement, which is calculated in the upper-level GA using the two-dimensional commonality chromosome and passed to all lower-level GAs during the fitness computation phase in each generation. In the upper-level GA, the first objective function minimizes the deviation from performance targets for the entire product family:

$$f_1 = \frac{1}{n_p} \left( \sum_{i=1}^{n_p} f'(i) \right) \tag{6}$$

where $f'_i$ is the first objective function received from the $i^{th}$ product (Eq. 5). The second objective is to maximize the commonality degree among products as defined by Eq. (2). Results follow.

## B. Results and Discussion

The GAA problem was solved with the proposed approach using both non-decomposed and decomposed implementations. In both cases, the following values for the multi-objective GA are used: population size = 200, maximum number of generations = 150, crossover probability = 0.9, and mutation probability = 0.15. The Pareto optimal front for the non-decomposed GA after 150 generations is shown in Figure 4a, where both objectives are plotted to be maximized. As can be seen in the figure, the non-decomposed algorithm has difficulty locating a set of diverse and well-converged points. This is due in part to the fact that adding the two-dimensional chromosome for generalizing commonality increases the problem complexity considerably.

The results from solving the same problem with the decomposed GA for 150 generations are shown in Figure 4b. Four processors were used for solving the GAA problem: one for the upper-level GA that controls commonality and three for the lower-level GAs, each optimizing an individual aircraft. The MPI (Message Passing Interface) library was used for sending and receiving data among the processors. As seen in the figure, a set of well-distributed points with various degrees of commonality and high performance are found. Comparing this figure with Figure 4a reveals that decomposing the original all-in-one algorithm improves the Pareto curve considerably and makes the optimization algorithm more scalable – the GA is no longer burdened by the lengthy chromosome string that results from treating the entire product family "all-at-once". In fact, because each GA was run for the same number of generations, the improvement shown in Figure 4 is due entirely to decomposition and does not show the additional reduction in computational time that can be attained with parallel computing. Moreover, the Pareto curve captures the tradeoff between commonality and individual distinctiveness in the product family: Increasing the commonality from 0% to about 60% yields only small differences in terms of deviation; however, increasing the commonality above 60% causes performance to decrease very rapidly.
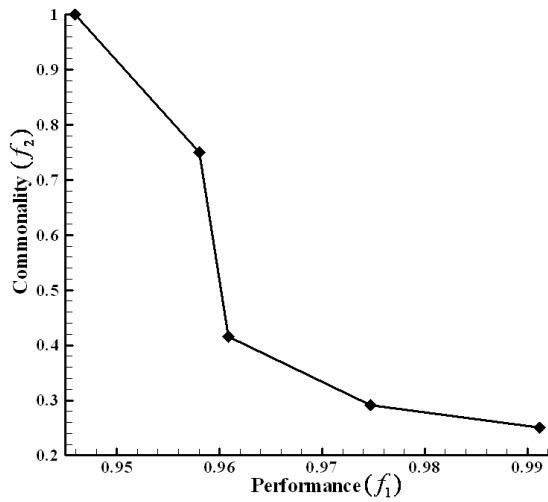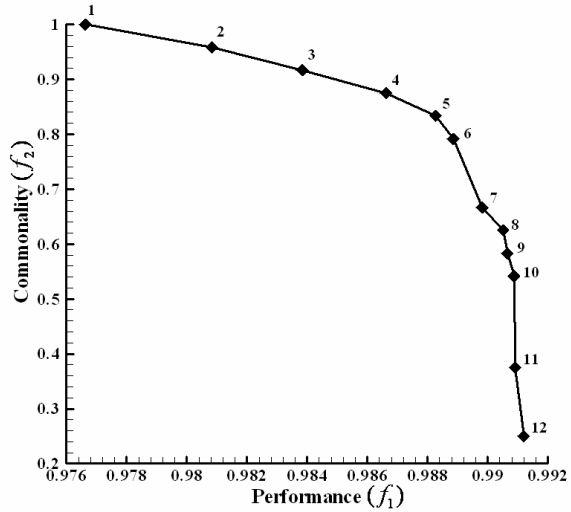
**4a. Non-Decomposed GA**                          **4b. Parallel GA**

**Figure 4. Comparison of Resulting Pareto Fronts for GAA Problem**

Design variables and deviation values for each point located in the Pareto optimal front (see Figure 4b) are listed in Table 4. The concept of generalized commonality can be seen from this table, i.e., all points (except the solution with 100% commonality) in the Pareto optimal front share variable values with a subset of the product family. The restricted approach of allowing only all-common or all-distinct is not capable of finding these solutions. This shows the effectiveness of the generalized two-dimensional commonality chromosome.

## VI.    Closing Remarks and Future Work

In this paper, we proposed a single-stage approach for solving the joint product family optimization problem using a novel decomposed MOGA formulation with a generalized commonality chromosome. The augmented chromosome representation introduced by Simpson and D'souza[7] was generalized to address component sharing among various subsets of products. Next, in order to improve the scalability of the proposed approach, the original all-in-one MOGA was decomposed into a two-level optimization problem in which the upper-level GA finds the optimal platform configuration while each lower-level GA optimizes one individual product in the family. A family of three general aviation aircraft was optimized using both the all-in-one (non-decomposed) and decomposed formulations for an equal number of generations. The Pareto optimal front found by the parallel algorithm is both well-converged and well-distributed along the entire commonality region, and it dominates all solutions of the all-in-one algorithm. Moreover, existence of component-sharing among various subsets of individual products in the optimal families reveals the importance of generalizing the commonality metric. Future work entails applying the approach to a larger product family to further test scalability and measuring the speedup due to both decomposition and parallel processing. Moreover, a more systematic way for generating the generalized commonality chromosome that ensures a proper distribution for the commonality probability density function should be investigated. This feature becomes critical in finding a well-distributed Pareto front when a family with a large number of products is considered.

**Table4. Design Variables and Deviation Values for Pareto Frontier in Figure 5b**

| Variables | Point 1 2-seat | 4-seat | 6-seat | Point 2 2-seat | 4-seat | 6-seat | Point 3 2-seat | 4-seat | 6-seat |
|---|---|---|---|---|---|---|---|---|---|
| WGS | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 |
| EMCRU | 9.69 | 9.69 | 9.69 | 9.21 | 9.21 | 9.21 | 9.77 | 9.77 | 9.77 |
| WS | 5.20 | 5.20 | 5.20 | 5.08 | 5.08 | 5.08 | 5.20 | 5.20 | 5.20 |
| WAS | 24.40 | 24.40 | 24.40 | 24.61 | 24.61 | 24.61 | 24.28 | 24.28 | 24.28 |
| AR | 95.49 | 95.49 | 95.49 | 91.36 | 91.36 | 91.36 | 85.78 | 85.78 | 85.78 |
| SAH | 17.60 | 17.60 | 17.60 | 15.45 | 17.22 | 17.22 | 17.45 | 18.64 | 18.64 |
| TCR | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| TCT | 17.10 | 17.10 | 17.10 | 18.39 | 18.39 | 18.39 | 19.51 | 19.51 | 19.51 |
| TCHT | 0.14 | 0.14 | 0.14 | 0.10 | 0.10 | 0.10 | 0.14 | 0.14 | 0.14 |
| ELODT | 0.12 | 0.12 | 0.12 | 0.18 | 0.18 | 0.18 | 0.12 | 0.12 | 0.12 |
| DPROP | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.15 | 0.08 | 0.08 |
| AF | 3.99 | 3.99 | 3.99 | 3.88 | 3.88 | 3.88 | 4.00 | 4.00 | 4.00 |
| Deviation(%) | 2.0594 | 1.5953 | 3.3556 | 1.6340 | 1.1539 | 2.9594 | 1.0380 | 0.9957 | 2.8142 |
| Commonality | 100% | | | 95.8% | | | 91.7% | | |

| Variables | Point 4 2-seat | 4-seat | 6-seat | Point 5 2-seat | 4-seat | 6-seat | Point 6 2-seat | 4-seat | 6-seat |
|---|---|---|---|---|---|---|---|---|---|
| WGS | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 |
| EMCRU | 9.74 | 9.74 | 9.74 | 9.77 | 9.77 | 9.77 | 9.45 | 9.45 | 9.98 |
| WS | 5.19 | 5.19 | 5.19 | 5.20 | 5.20 | 5.20 | 5.13 | 5.13 | 5.35 |
| WAS | 24.99 | 24.99 | 24.99 | 24.29 | 24.29 | 24.29 | 24.61 | 24.61 | 24.61 |
| AR | 95.91 | 95.91 | 95.91 | 85.66 | 85.66 | 85.66 | 91.78 | 91.78 | 91.78 |
| SAH | 16.35 | 17.66 | 19.61 | 17.45 | 18.68 | 18.68 | 15.74 | 17.19 | 19.02 |
| TCR | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| TCT | 17.77 | 17.77 | 17.77 | 19.56 | 19.56 | 19.56 | 17.81 | 17.81 | 17.81 |
| TCHT | 0.11 | 0.11 | 0.11 | 0.14 | 0.14 | 0.10 | 0.10 | 0.10 | 0.10 |
| ELODT | 0.18 | 0.18 | 0.18 | 0.12 | 0.12 | 0.18 | 0.18 | 0.18 | 0.18 |
| DPROP | 0.15 | 0.08 | 0.08 | 0.15 | 0.08 | 0.08 | 0.15 | 0.08 | 0.08 |
| AF | 3.95 | 3.95 | 3.95 | 4.00 | 4.00 | 4.00 | 4.00 | 4.00 | 4.00 |
| Deviation(%) | 1.0658 | 0.8636 | 2.0841 | 1.0230 | 1.0170 | 1.4700 | 1.0390 | 0.8880 | 1.6040 |
| Commonality | 87.5% | | | 83.3% | | | 79.2% | | |

| Variables | Point 7 2-seat | 4-seat | 6-seat | Point 8 2-seat | 4-seat | 6-seat | Point 9 2-seat | 4-seat | 6-seat |
|---|---|---|---|---|---|---|---|---|---|
| WGS | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 |
| EMCRU | 9.75 | 9.75 | 10.00 | 9.08 | 9.54 | 10.00 | 9.07 | 9.47 | 10.00 |
| WS | 5.19 | 5.19 | 5.40 | 5.08 | 5.08 | 5.49 | 5.06 | 5.19 | 5.40 |
| WAS | 24.98 | 24.98 | 24.55 | 24.94 | 24.55 | 24.94 | 24.95 | 24.57 | 24.82 |
| AR | 95.88 | 95.88 | 98.05 | 86.19 | 86.19 | 86.19 | 89.03 | 89.03 | 89.03 |
| SAH | 16.34 | 17.62 | 19.31 | 15.26 | 17.16 | 19.07 | 15.26 | 17.15 | 19.02 |
| TCR | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| TCT | 18.53 | 17.50 | 18.53 | 17.30 | 18.50 | 19.0019.56 | 18.50 | 18.50 | 18.50 |
| TCHT | 0.11 | 0.11 | 0.11 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 |
| ELODT | 0.18 | 0.18 | 0.18 | 0.18 | 0.18 | 0.20 | 0.18 | 0.18 | 0.20 |
| DPROP | 0.15 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 |
| AF | 3.94 | 3.94 | 3.94 | 3.96 | 3.96 | 3.96 | 3.86 | 3.86 | 4.00 |
| Deviation(%) | 1.0761 | 0.8266 | 1.1504 | 0.9630 | 0.8210 | 1.0560 | 0.9314 | 0.8537 | 1.0150 |
| Commonality | 66.7% | | | 62.5% | | | 58.3% | | |

| Variables | Point 10 2-seat | 4-seat | 6-seat | Point 11 2-seat | 4-seat | 6-seat | Point 12 2-seat | 4-seat | 6-seat |
|---|---|---|---|---|---|---|---|---|---|
| WGS | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.24 | 0.25 |
| EMCRU | 9.07 | 9.54 | 10.00 | 9.81 | 9.82 | 10.00 | 9.08 | 9.79 | 9.68 |
| WS | 5.06 | 5.10 | 5.34 | 5.17 | 5.31 | 5.31 | 5.11 | 5.35 | 5.47 |
| WAS | 24.93 | 24.56 | 24.58 | 24.34 | 24.26 | 24.55 | 24.94 | 24.26 | 24.23 |
| AR | 87.03 | 87.03 | 87.03 | 87.65 | 82.60 | 90.00 | 87.00 | 83.03 | 80.13 |
| SAH | 15.26 | 17.15 | 18.95 | 17.31 | 18.68 | 18.92 | 15.43 | 18.67 | 19.07 |
| TCR | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| TCT | 18.56 | 19.06 | 19.48 | 17.92 | 18.20 | 17.92 | 18.08 | 19.00 | 18.15 |
| TCHT | 0.10 | 0.10 | 0.10 | 0.14 | 0.14 | 0.10 | 0.10 | 0.14 | 0.10 |
| ELODT | 0.18 | 0.18 | 0.19 | 0.12 | 0.12 | 0.19 | 0.18 | 0.12 | 0.18 |
| DPROP | 0.08 | 0.08 | 0.08 | 0.15 | 0.08 | 0.08 | 0.09 | 0.08 | 0.08 |
| AF | 3.99 | 3.99 | 3.99 | 3.98 | 3.98 | 3.96 | 3.91 | 3.98 | 3.97 |
| Deviation(%) | 0.9601 | 0.7774 | 0.9955 | 0.9210 | 0.8406 | 0.9625 | 0.8450 | 0.8040 | 0.9900 |
| Commonality | 54.2% | | | 37.5% | | | 25.0% | | |

## Acknowledgments

## References

[1]Meyer, M. H., 1997, "Revitalize Your Product Lines Through Continuous Platform Renewal," *Research Technology Management,* 40(2), pp. 17-28.

[2]Khire, R. A., Messac, A. and Simpson, T. W., 2006, "Optimal Design of Product Families Using Selection-Integrated Optimization (SIO) Methodology," 11th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Portsmouth, VA, AIAA-2006-6924.

[3]Simpson, T. W., 2005, "Methods for Optimizing Product Platforms and Product Families: Overview and Classification," *Product Platform and Product Family Design: Methods and Applications*, T. W. Simpson, Z. Siddique and J. Jiao, eds.*,* Springer, New York, pp. 133-156.

[4]Allada, V. and Jiang, L., 2002, "New Modules Launch Planning for Evolving Modular Product Families," *ASME Design Engineering Technical Conferences - Design for Manufacturing Conference,* Montreal, Quebec, Canada, ASME, Paper No. DETC2002/DFM-34190.

[5]Blackenfelt, M., 2000, "Profit Maximisation while Considering Uncertainty by Balancing Commonality and Variety Using Robust Design - The Redesign of a Family of Lift Tables," *ASME Design Engineering Technical Conferences - Design for Manufacturing,* Baltimore, MD, ASME, Paper No. DETC2000/DFM-14013.

[6]Chang, T.-S. and Ward, A. C., 1995, "Design-in-Modularity with Conceptual Robustness," *Advances in Design Automation,* Boston, MA, ASME, Vol. 82-1, pp. 493-500.

[7]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-Research and Applications,* 12(2), pp. 119-129.

[8]Dai, Z. and Scott, M. J., 2004, "Product Platform Design through Sensitivity Analysis and Cluster Analysis," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Salt Lake City, UT, ASME, Paper No. DETC2004/DAC-57464.

[9]Farrell, R. and Simpson, T. W., 2003, "Product Platform Design to Improve Commonality in Custom Products," *Journal of Intelligent Manufacturing,* 14(6), pp. 541-556.

[10]Fellini, R., Papalambros, P. and Weber, T., 2000, "Application of Product Platform Design Process to Automoative Powertrains," *8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Long Beach, CA, AIAA, AIAA-2000-4849.

[11]Fujita, K., Akagi, S., Yoneda, T. and Ishikawa, M., 1998, "Simultaneous Optimization of Product Family Sharing System Structure and Configuration," *ASME Design Engineering Technical Conferences - Design for Manufacturing,* Atlanta, GA, ASME, Paper No. DETC98/DFM-5722.

[12]Gonzalez-Zugasti, J. P., Otto, K. N. and Baker, J. D., 2000, "A Method for Architecting Product Platforms," *Research in Engineering Design,* 12(2), pp. 61-72.

[13]Gonzalez-Zugasti, J. P., Otto, K. N. and Baker, J. D., 2001, "Assessing Value for Platformed Product Family Design," *Research in Engineering Design,* 13(1), pp. 30-41.

[14]Hernandez, G., Simpson, T. W., Allen, J. K., Bascaran, E., Avila, L. F. and Salinas, F., 2001, "Robust Design of Families of Products with Production Modeling and Evaluation," *ASME Journal of Mechanical Design,* 123(2), pp. 183-190.

[15]Kokkolaras, M., Fellini, R., Kim, H. M., Michelena, N. F. and Papalambros, P. Y., 2002, "Extension of the target cascading formulation to the design of product families," *Structural and Multidisciplinary Optimization,* 24(4), pp. 293-301.

[16]Kumar, R., Allada, V. and Ramakrishnan, S., 2004, "Ant Colony Optimization Methods for Product Platform Formation," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Salt Lake City, UT, ASME, Paper No. DETC2004/DAC-57195.

[17]Li, H. and Azarm, S., 2002, "An Approach for Product Line Design Selection Under Uncertainty and Competition," *ASME Journal of Mechanical Design,* 124(3), pp. 385-392.

[18]Messac, A., Martinez, M. P. and Simpson, T. W., 2002, "Effective Product Family Design Using Physical Programming," *Engineering Optimization,* 34(3), pp. 245-261.

[19]Nelson, S. A., II, Parkinson, M. B. and Papalambros, P. Y., 2001, "Multicriteria Optimization in Product Platform Design," *ASME Journal of Mechanical Design,* 123(2), pp. 199-204.

[20]Ortega, R., Kalyan-Seshu, U. and Bras, B., 1999, "A Decision Support Model for the Life-Cycle Design of a Family of Oil Filters," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Las Vegas, NV, ASME, Paper No. DETC99/DAC-8612.

[21]Seepersad, C. C., Hernandez, G. and Allen, J. K., 2000, "A Quantitative Approach to Determining Product Platform Extent," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Baltimore, MD, ASME, Paper No. DETC2000/DAC-14288.

[22]Seepersad, C. C., Mistree, F. and Allen, J. K., 2002, "A Quantitative Approach for Designing Multiple Product Platforms for an Evolving Portfolio of Products," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Montreal, Quebec, Canada, ASME, Paper No. DETC2002/DAC-34096.

[23]Simpson, T. W., Maier, J. R. A. and Mistree, F., 1999, "A Product Platform Concept Exploration Method for Product Family Design," *Design Theory and Methodology - DTM'99,* Las Vegas, Nevada, ASME, Paper No. DETC99/DTM-8761.

[24]Simpson, T. W., Maier, J. R. A. and Mistree, F., 2001, "Product Platform Design: Method and Application," *Research in Engineering Design,* 13(1), pp. 2-22.

[25]Akundi, S., Simpson, T. W. and Reed, P. M., 2005, "Multi-objective Design Optimization for Product Platform and Product Family Design Using Genetic Algorithms," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Long Beach, CA, ASME, Paper No. DETC2005/DAC-84905.

[26]Cetin, O. L. and Saitou, K., 2004, "Decomposition-Based Assembly Synthesis for Structural Modularity," *ASME Journal of Mechanical Design,* 126(2), pp. 234-243.

[27]de Weck, O., Suh, E. S. and Chang, D., 2003, "Product Family and Platform Portfolio Optimization," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Chicago, IL, ASME, Paper No. DETC2003/DAC-48721.

[28]Fellini, R., Kokkolaras, M., Michelena, N., Papalambros, P., Saitou, K., Perez-Duarte, A. and Fenyes, P. A., 2002, "A Sensitivity-Based Commonality Strategy for Family Products of Mild Variation, With Application to Automotive Body Structures," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Atlanta, GA, AIAA, AIAA-2002-5610.

[29]Fellini, R., Kokkolaras, M., Papalambros, P. and Perez-Duarte, A., 2002, "Platform Selection Under Performance Loss Constraints in Optimal Design of Product Families," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Montreal, Quebec, Canada, ASME, Paper No. DETC2002/DAC-34099.

[30]Fujita, K. and Yoshida, H., 2001, "Product Variety Optimization: Simultaneous Optimization of Module Combination and Module Attributes," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Pittsburgh, PA, ASME, Paper No. DETC2001/DAC-21058.

[31]Fujita, K., Sakaguchi, H. and Akagi, S., 1999, "Product Variety Deployment and Its Optimization Under Modular Architecture and Module Commonalization," *ASME Design Engineering Technical Conferences - Design for Manufacturing,* Las Vegas, NV, ASME, Paper No. DETC99/DFM-8923.

[32]Gonzalez-Zugasti, J. P. and Otto, K. N., 2000, "Modular Platform-Based Product Family Design," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Baltimore, MD, ASME, Paper No. DETC-2000/DAC-14238.

[33]Hernandez, G., Allen, J. K. and Mistree, F., 2002, "Design of Hierarchic Platforms for Customizable Products," *ASME Design Engineering Technical Conferences - Design Automation Conference,* Montreal, Quebec, Canada, ASME, Paper No. DETC2002/DAC-34095.

[34]Hernandez, G., Allen, J. K. and Mistree, F., 2003, "Platform Design for Customizable Products as a Problem of Access in a Geometric Space," *Engineering Optimization,* 35(3), pp. 229-254.

[35]Messac, A., Martinez, M. P. and Simpson, T. W., 2002, "A Penalty Function for Product Family Design Using Physical Programming," *ASME Journal of Mechanical Design,* 124(2), pp. 164-172.

[36]Nayak, R. U., Chen, W. and Simpson, T. W., 2002, "A Variation-Based Method for Product Family Design," *Engineering Optimization,* 34(1), pp. 65-81.

[37]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,* 41(17), pp. 4075-4098.

[38]Hassan, R., de Weck, O. and Springmann, P., 2004, "Architecting a Communication Satellite Product Line," *22nd AIAA International Communications Satellite Systems Conference & Exhibit 2004 (ICSSC),* Monterey, CA, AIAA, AIAA-2004-3150.

[39]Fujita, K., 2002, "Product Variety Optimization Under Modular Architecture," *Computer-Aided Design,* 34(12), pp. 953-965.

[40]D'Souza, B. and Simpson, T. W., 2003, "A Genetic Algorithm Based Method for Product Family Design Optimization," *Engineering Optimization,* 35(1), pp. 1-18.

[41]Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T., 2000, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II," *Parallel Problem Solving from Nature VI Conference,* Paris, France, pp. 849-858.

[42]Khajavirad, A. and Michalek, J., 2007, "An extension of the commonality index for product family optimization,", in review,ASME 2007 International Design Engineering Technical Conferences, Las Vegas, Nevada, USA.

[43]Martin, M. and Ishii, K., 1996, "Design for Variety: A Methodology for Understanding the Costs of Product Proliferation," *Design Theory and Methodology - DTM'96,* Irvine, CA, ASME, Paper No. 96-DETC/DTM-1610.

[44]Simpson, T. W., Chen, W., Allen, J. K. and Mistree, F., 1996, "Conceptual Design of a Family of Products Through the Use of the Robust Concept Exploration Method," *6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Bellevue, WA, AIAA, Vol. 2, pp. 1535-1545.