Struct Multidisc Optim (2009) 39:187–201
DOI 10.1007/s00158-008-0321-9

INDUSTRIAL APPLICATION

An efficient decomposed multiobjective genetic algorithm for solving the joint product platform selection and product family design problem with generalized commonality

Aida Khajavirad · Jeremy J. Michalek · Timothy W. Simpson

Received: 10 June 2007 / Revised: 11 August 2008 / Accepted: 22 September 2008 / Published online: 5 November 2008

Abstract Product family optimization involves not only specifying the platform from which the individual product variants will be derived, but also optimizing the platform design and the individual variants. Typically these steps are performed separately, but we propose an efficient decomposed multiobjective genetic algorithm to jointly determine optimal (1) platform selection, (2) platform design, and (3) variant design in product family optimization. The approach addresses limitations of prior restrictive component sharing definitions by introducing a generalized two-dimensional commonality chromosome to enable sharing components among subsets of variants. To solve the resulting high dimensional problem in a single stage efficiently, we exploit the problem structure by decomposing it into a two-level genetic algorithm, where the upper level determines the optimal platform configuration while each lower level optimizes one of the individual variants. The decomposed approach improves scalability of the all-in-one problem dramatically, providing a practical tool for optimizing families with more variants. The proposed approach is demonstrated by optimizing a family of electric motors. Results indicate that (1) decomposition results in improved solutions under comparable computational cost and (2) generalized commonality produces families with increased component sharing under the same level of performance.

Keywords Product family · Optimization · Genetic algorithms · Decomposition · Generalized commonality

A preliminary version of this paper was presented at the 2007 AIAA Multidisciplinary Design Optimization Specialists Conference.

A. Khajavirad
Mechanical Engineering, Carnegie Mellon University,
Pittsburgh, PA 15213, USA
e-mail: aida@cmu.edu

J. J. Michalek (✉)
Mechanical Engineering and Engineering and Public Policy,
Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: jmichalek@cmu.edu

T. W. Simpson
Mechanical and Industrial Engineering,
The Pennsylvania State University,
University Park, PA 16802, USA
e-mail: tws8@psu.edu

1 Introduction

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 called *product platforms* to satisfy a variety of market niches. Designing a 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 variants without drastically compromising their individual performance (Simpson et al. 2001). This challenge manifests early in the design process wherein designers must not only specify the *platform configuration*—also referred to as *platform variable selection* or *platform selection* (Khire et al. 2006)—but also optimize the design of the platform as well as the individual variants derived from the platform.

Resolving the inherent tradeoff between platform commonality and distinct variant performance is paramount: Increasing the degree of commonality among products generally reduces total cost, but it can also compromise the ability of each variant to fully achieve the desired characteristics making it distinct and attractive to different market segments. The broad problem of product family design involves many issues, such as supply chain management, manufacturing investment, estimation of cost structures, and market positioning (de Weck 2005). We focus here on developing an improved optimization algorithm for solving the joint product family platform selection and design problem by mapping the tradeoff between increased component commonality among variants vs. achievement of distinct, exogenous performance targets for each product. The aforementioned issues could be integrated within the proposed framework through appropriate modification of objective functions and problem formulation. In the next section, we review related optimization-based research that has sought to address this tradeoff, and in Section 3 we describe our novel multiobjective genetic algorithm (MOGA)-based approach for solving the joint product family platform-selection and variant design problem with generalized commonality. In Section 4, the structure of the product family problem is exploited to decompose the all-in-one formulation into a two-level MOGA, which improves its search efficiency and scalability dramatically by reducing the search space of each sub-GA and enabling use of parallel processing. In Section 5, an example involving the design of a family of electric motors is presented and optimized, and the effects of decomposition and commonality generalization and also the complexity of the proposed method with respect to the number of variants are investigated. Closing remarks and future work are discussed in Section 6.

2 Review of related literature

2.1 Classification: product family optimization

Numerous optimization approaches have been developed within the engineering design community during the past decade to solve the product family design problem. Simpson (2005) reviews and classifies forty such approaches from the literature. In many of these approaches, product platforms are known or specified a priori, i.e., before performing the optimization, whereas in other instances, platform-selection is determined during optimization (i.e., the platform is specified a posteriori.) In a similar manner, Fujita (2002) classified product family optimization problems into three classes (see Fig. 1): In **Class I** problems (boxes 1 and 2), product attributes are optimized under a fixed platform configuration (i.e., the platform is known a priori); **Class II** problems (boxes 3 and 4) find the optimal module selection from predefined sets of modules (i.e., the design of each module is known a priori); and finally, in **Class III** problems (boxes 5 and 6) the product attributes and platform configuration are optimized simultaneously. It is this Class III, a posteriori problem that we refer to as the *joint product family platform selection and design problem* (or the *joint problem* for short) in that it involves determining the optimal combination of 1) *platform variable selection*, 2) *platform design* and 3) *variant design*. Each of these decisions is generally dependent on the others (it is typically not possible to know the optimal platform without first knowing the variant design, and vise versa); so, Class I and Class II problems cannot generally offer optimality with respect to the full problem. Thus, we focus our attention on approaches to address the Class III joint problem.

In Fig. 1, the classification of methods is further refined by adding the dimension of **restricted vs. generalized commonality**: Methods that employ *restricted commonality* limit the commonality definition to all-or-none component sharing; that is, a component can either be common within the entire family or be distinct among all variants. A *generalized commonality* formulation avoids this restriction and allows for component sharing within subsets of the variants. The restricted definition is a simplifying assumption that is typically employed to decrease computational complexity; however, it imposes significant limitations that are often not observed in product family design practice (Thevenot and Simpson 2006). Therefore, there is a need for an

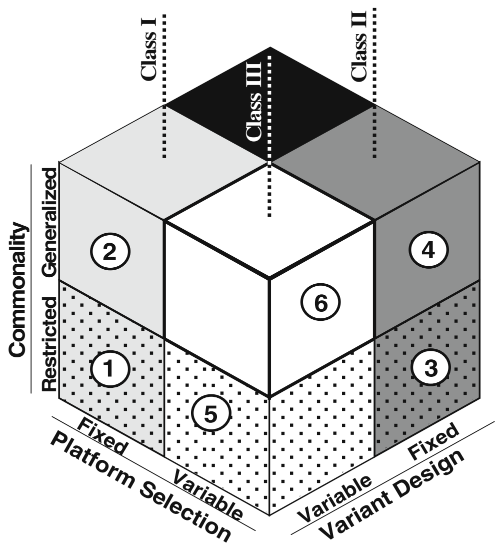


Fig. 1 Classification of product family optimization formulations

approach capable of solving the joint problem using generalized commonality (box 6) for practical product family applications with a reasonable computational cost.

2.2 Prior approaches to solving the joint problem

Most of the previous *a posteriori* optimization methods reviewed by Simpson (2005) avoid the high computational cost of the joint problem by dividing it into multiple stages; that is, instead of addressing a Class III problem directly, in the first stage the platform configuration is found followed by a Class I problem to find the variant design using the fixed platform found in the first stage. However, the two-stage approaches have been shown to lead to suboptimal solutions (Messac et al. 2002); therefore, single-stage approaches are preferred for optimality.

Single-stage Class III problems typically employ commonality restrictions to reduce computational cost (box 5) (Simpson and D'Souza 2004; Hassan et al. 2004; Khire et al. 2006), and therefore, suffer from suboptimality. Fujita and Yoshida (2001) addressed generalized commonality for the joint problem (box 6) by hybridizing GA, branch and bound, and sequential quadratic programming (SQP) to determine platform configuration, direction of similarities and variant design respectively. The approach may be well-suited to convex problems; however, for non-convex problems, it may generate suboptimal solutions due to local search in the fitness evaluation. Moreover, the nesting of optimization algorithms leads to high computational costs. Khajavirad and Michalek (2008a) proposed a decomposed approach that relaxes the combinatorial platform selection variables to the continuous space and solves the generalized joint problem (box 6) through a sequence of relaxations. While the proposed method is computationally efficient, it suffers from the same problem of suboptimality for nonconvex problems due to combining a heuristic (relaxation to continuous space) and a local search optimization method.

In brief, a single-stage optimization approach for solving the joint product family platform-selection and design problem with generalized commonality (box 6) is needed that can solve practical problems under reasonable computational costs without employing local search methods that assume problem convexity.

2.3 Decomposition approaches

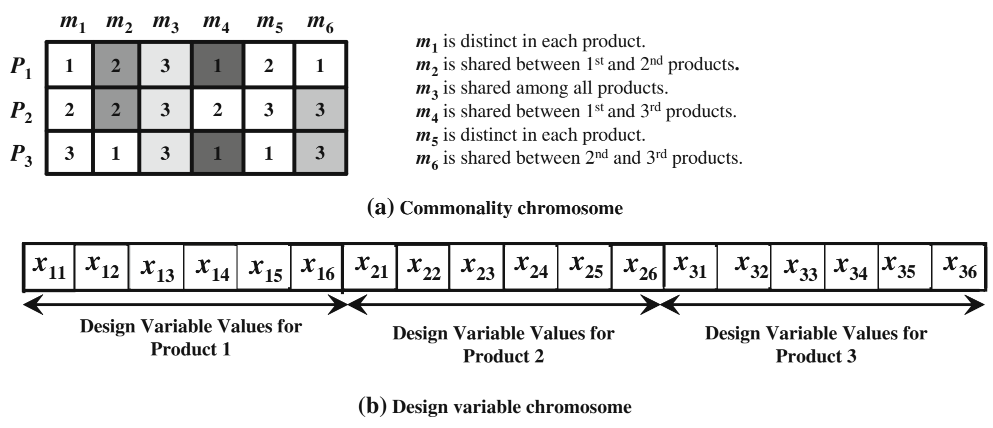
An approach to reducing computational cost of the optimization problems with special structures is to decompose the all-in-one formulation into a set of interrelated sub-problems such that solving and coordinating the individual sub-problems is faster or more robust than optimizing the full problem all-in-one. The hierarchical structure of product families can be exploited in this way. Fujita (2002) decomposed the Class III product family optimization problem into module combination and module attribute optimization sub-problems, and solved sub-problems in nested loops. Kokkolaras et al. (2002), applied analytical target cascading (ATC) for decomposing a Class I problem by allocating each individual product design to a separate sub-problem and imposing commonality decisions by introducing subsystems with multiple parents. Michalek et al. (2006a, b) also applied ATC to decompose a line of products including market demand and manufacturing data; however, the approach considered only manufacturing equipment sharing and did not allow for component commonality among variants. Finally, Khajavirad and Michalek (2008a) introduced a two-level ATC-based decomposition scheme for solving the joint problem through a sequence of continuous relaxations; however, the approach is intended for convex formulations where design variables are continuous and gradients are available. Hence, a decomposition approach based on a non-gradient global search algorithm could avoid assumptions of convexity, continuity, or gradient availability, broadening the scope of applicability for many practical product family problems.

3 Proposed MOGA approach

In this paper, we introduce a powerful new MOGA formulation for determining the tradeoff between platform commonality and individual variant performance in the class III joint product family problem with generalized commonality. The underlying algorithm for our MOGA code is the elitist non-dominated sorting GA (NSGA-II) introduced by Deb et al. (2000) 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. However, in order to apply the original NSGA-II code to the joint problem, we have modified the chromosome representation, crossover, and mutation operators as described in the sections that follow.

3.1 Chromosome representation

We generalize the augmented chromosome representation of Simpson and D'Souza (2004) to relax the all-or-none component sharing restriction so that platform variables can be shared among any subset of



m_1 m_2 m_3 m_4 m_5 m_6
P_1 1 2 3 1 2 1
P_2 2 2 3 2 3 3
P_3 3 1 3 1 1 3
m_1 is distinct in each product.
m_2 is shared between 1st and 2nd products.
m_3 is shared among all products.
m_4 is shared between 1st and 3rd products.
m_5 is distinct in each product.
m_6 is shared between 2nd and 3rd products.
(a) Commonality chromosome
x_{11} x_{12} x_{13} x_{14} x_{15} x_{16} x_{21} x_{22} x_{23} x_{24} x_{25} x_{26} x_{31} x_{32} x_{33} x_{34} x_{35} x_{36}
Design Variable Values for Product 1
Design Variable Values for Product 2
Design Variable Values for Product 3
(b) Design variable chromosome

Fig. 2 Chromosome representation for each product family in the MOGA population (p_i: i^{th} product, m_j: j^{th} component, x_{ij}: j^{th} component of i^{th} product)

variants. This generalization is achieved by introducing two parallel chromosomes for each individual in the MOGA population (see Fig. 2): 1) a two-dimensional *commonality chromosome* that defines the platform configuration and allows for component sharing among subsets of products, and 2) a one-dimensional *design variable chromosome* that contains design variables of all variants in the family. Hence, in a product family with p products, each defined by n components,[1] the commonality chromosome is a two-dimensional matrix with p rows and n columns, and 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 equal integer values indicate that the corresponding components are common.[2] An example of this representation for a product family with three products and six components is shown in Fig. 2.

3.2 Consistency constraints

The proposed algorithm ensures that the two chromosomes remain consistent during the evolution using *consistency constraints*, which are classified into two groups: design consistency and commonality consistency. The design consistency constraints ensure that the design variables are consistent with the commonality chromosome: For each set of components identified as common by the commonality chromosome, the corresponding design variable genes are replaced by the average value[3] within the set (see Fig. 3a). The commonality consistency constraint ensures that the commonality chromosome is consistent with the design chromosomes at each iteration: If the component genes' values for any subset of products differ by less than the maximum user-defined tolerance[4] as a result of the crossover or mutation operators, the corresponding commonality genes are modified accordingly (see Fig. 3b).[5]

3.3 Crossover operators

Due to the 2-D configuration of the commonality chromosome, a *two-dimensional binary crossover operator* was applied, which is a direct extension of the one-point crossover operator to two dimensions. In this operator, two random integer 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 components in each product, respectively. These two random numbers are used to divide the commonality chromosome into four quadrants. Then, a third random

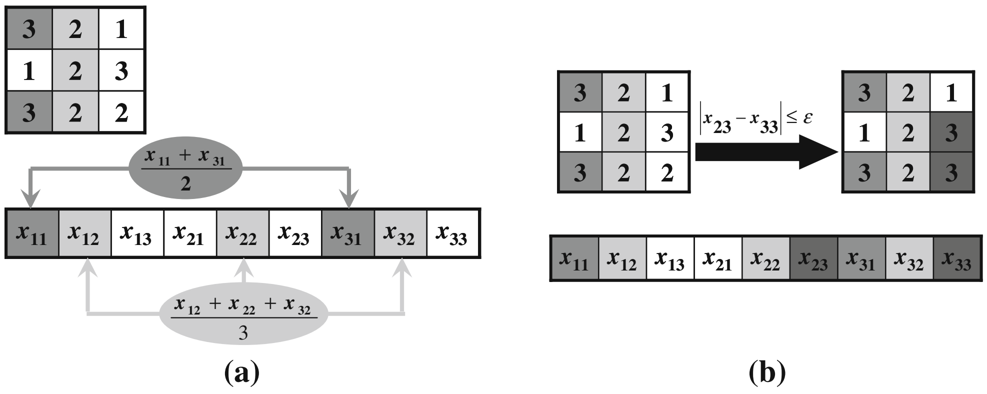
[1]Here, without loss of generality, we assume each component is represented by a single design variable (i.e. gene); however, the algorithm is applicable to the case which each component includes different number of design variables.

[2]In our discussion, we assume that all products include the same number of components as candidates for commonality. However, this representation can be modified to include a general case in which any subset of components may be absent in a variant by setting the corresponding gene in the 2D chromosome to a distinct integer number (e.g. zero) and omitting those genes from the design variable chromosome of that variant. In addition, for the components that are only present in $p' < p$ products, their commonality genes can take any integer value between 1 and p'.

[3]In case of discrete variables, the average value should be further rounded to the closest discrete level. Moreover, this constraint can be imposed using other strategies such as generating a random number in the upper level and sending it to the lower levels.

[4]The user-defined tolerance for considering two design variables to be equal should be set using knowledge about the problem, including the physical interpretation of the variable values and sensitivity of performance to the value of these variables. Thus, setting of the user-tolerance is necessarily case-specific.

[5]It should be noted that the commonality consistency constraints are only imposed for finding the optimal solution faster, and the method can identify optimal solutions without these constraints as well.



$\frac{x_{11}+x_{31}}{2}$
$\frac{x_{12}+x_{22}+x_{32}}{3}$
$|x_{23}-x_{33}|\le\varepsilon$
(a)
(b)

Fig. 3 Consistency constraints: (**a**) design consistency and (**b**) commonality consistency

integer, in the range of [1, 4], is generated to decide which quadrant is to be interchanged (see Fig. 4).

The crossover type applied to the design variable chromosome is the default operator used in the original NSGA-II code, which is *simulated binary crossover* (Deb 2001).

3.4 Mutation operators

The mutation operator is designed to mutate the platform configuration of the product family to enhance the searching quality of the GA for exploring various levels of commonality. First, for each component, a random number is generated ($0 \le rnd_1 \le 1$). If its value is less than the mutation probability (p_m), then the corresponding component is mutated. In mutation, a new random number is generated ($0 \le rnd_2 \le 1$). If its value is less than 0.5, then that component is set as distinct in each product, and is mutated according to the *polynomial mutation operator*. Otherwise, the component is made common among all products by first being mutated in each variant and then being replaced by its average value followed by a rounding strategy in case of discrete variables (see Fig. 5). After applying crossover and mutation operators, the algorithm modifies both chromosomes according to the consistency constraints.

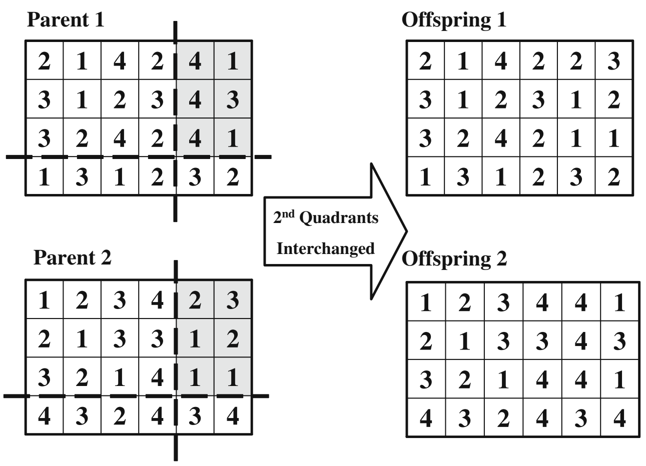


Fig. 4 Two-dimensional binary crossover operator

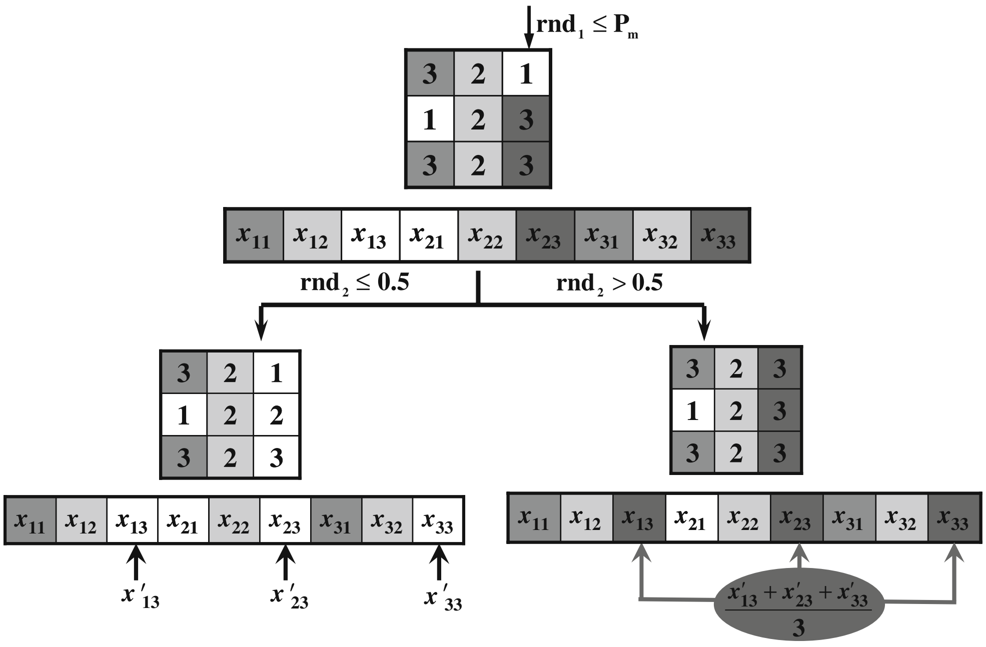
3.5 Commonality objective function

In order to have the MOGA find the optimal platform configuration, an objective function for measuring the commonality of 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. Khajavirad and Michalek (2008a) argue that the commonality index (CI), introduced by Martin and Ishii (1996), is a better metric for measuring tooling cost savings of component sharing relative to prior metrics used in product family optimization,[6] 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:

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

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} component in the

[6]To estimate the tooling cost savings more precisely, CI should be reformulated to include coefficients representing the amount of cost saving due to sharing each component. However, since this extension does not affect the optimization approach, all coefficients are assumed to be equal in this paper.



$\text{rnd}_1 \le \text{P}_m$
x_{11} x_{12} x_{13} x_{21} x_{22} x_{23} x_{31} x_{32} x_{33}
$\text{rnd}_2 \le 0.5$
$\text{rnd}_2 > 0.5$
x'_{13}
x'_{23}
x'_{33}
$\frac{x'_{13}+x'_{23}+x'_{33}}{3}$

Fig. 5 Mutation operators ($x'_{ij} = f(x_{ij})$, f polynomial mutation operator, x'_{ij} mutated value of x_{ij})

commonality chromosome, (1) can be reformulated as follows:

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

Using this definition, the commonality objective function can be calculated using only the commonality chromosome while the product performance-related objectives are evaluated using each design variable chromosome; this is the key feature that enables decomposition of the proposed MOGA, as discussed next.

4 Decomposition and parallelization of the MOGA

Aforementioned modifications to the original NSGA-II code make it convenient for optimizing the joint product family problem with generalized commonality; however, this algorithm is still only practical for problems with a relatively small number of components and variants. The commonality generalization also increases this complexity, making the all-in-one algorithm inefficient in dealing with high-dimensional problems. To address this scalability limitation, we propose a decomposition of the original formulation (see Fig. 6). The new 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. In addition, the consistency constraints are imposed to all sub-problems.

The general structure of the proposed model is shown in Fig. 7. The steps of the algorithm proceed as follows:

1. Initialization: Initial populations are generated in the upper-level and lower-level GAs independently. Next, according to the consistency constraints both commonality and design variable chromosomes send the required data[7] to lower- and upper-level GAs, respectively.
2. Fitness Calculation: The commonality metric, (2), and individual variant performance objectives are calculated in the upper- and lower-level GAs, respectively. The upper-level GA sends the commonality metric of each population member to all lower-level GAs, which are included in the fitness function of the corresponding individual in each sub-GA in addition to the product performance objectives. Each lower-level GA also returns performance deviations to the upper-level GA, which are then summed across variants to form the overall performance objective functions.
3. Crossover and Mutation: Using the aforementioned crossover and mutation operators, offspring populations are generated in all GAs independently. Crossover and mutation operators at each sub-problem are the same as the sequential version except that the tasks are divided among differ-

[7]Data exchange necessary to enforce consistency constraints is handled through Message Passing Interface (MPI) library. Details of the implementation and a copy of the code are available through the authors or at http://www.cmu.edu/me/ddl.

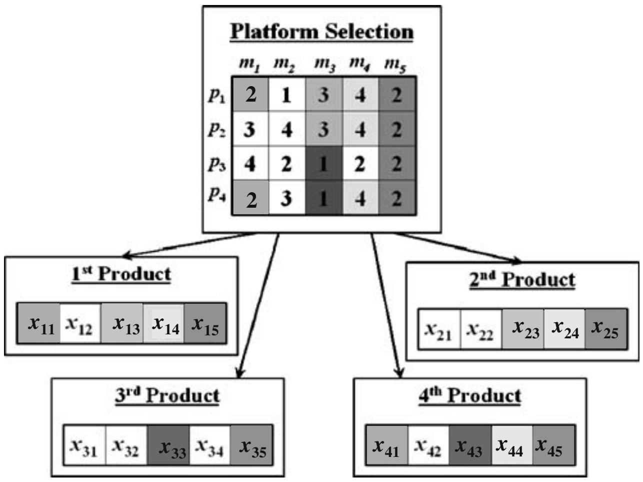


Fig. 6 Allocating the two parallel chromosome representation to a two-level MOGA (p_i: i^{th} product, m_j: j^{th} component, x_{ij}: j^{th} component of the i^{th} product)

ent 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. In case of mutation, the upper-level GA determines which components should be 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. After all offspring populations are generated, they are modified according to the consistency constraints, and each lower-level GA passes back the fitness value for its corresponding performance objective(s).

4. Replacement: The upper-level GA combines the parent and offspring population and applies non-dominated sorting with respect to the commonality level and overall performance objectives to select the best half as the new generation that will define the new population in all (upper- and lower-level) GAs.
5. Iteration and Termination: If the generation number is equal to the maximum generation number, then the algorithm is terminated; otherwise, the process is repeated from Step 2.

Using the proposed decomposition scheme, the dimensionality of each lower-level GA remains constant regardless of the number of variants in the product family; this is the key feature making the decomposed approach scalable. However, this improved scalability is still limited by the size of commonality chromosome in that it grows linearly as number of variants increases; these desirable and adverse effects are further quantified in section 5–5. In the all-in-one GA, selection of product families from the population is made with respect to the overall fitness value of those families; in contrast, the decomposed GA involves (1) selection of design variable chromosomes based on

Fig. 7 Decomposed MOGA model for product family design

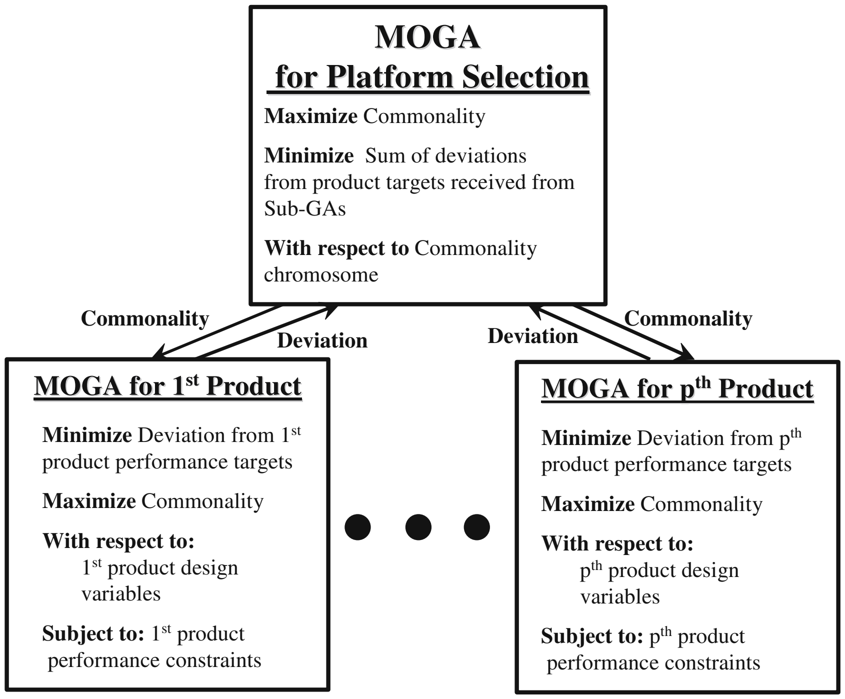

MOGA for Platform Selection
Maximize Commonality
Minimize Sum of deviations from product targets received from Sub-GAs
With respect to Commonality chromosome
Commonality
Deviation
Deviation
Commonality
MOGA for 1st Product
Minimize Deviation from 1st product performance targets
Maximize Commonality
With respect to: 1st product design variables
Subject to: 1st product performance constraints
MOGA for pth Product
Minimize Deviation from pth product performance targets
Maximize Commonality
With respect to: pth product design variables
Subject to: pth product performance constraints

Table 1 Design variables and bounds for electric motor example

Definition	Lower bound	Upper bound
Wire turns on the armature: *Nc*	100	1,500
Wire turns on each field pole: *Ns*	1	500
Armature wire area (mm^2): *Awa*	0.01	1.00
Field wire area (mm^2): *Awf*	0.01	1.00
Radius of the motor (mm): r_o	10	100.0
Thickness of the stator (mm): t	0.50	100.0
Stack length of the motor (mm): L	1.0	100.0
Current drawn by the motor	0.1	6.0

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. The commonality value for the entire family is also included as an additional objective function for each sub-GA which is only used for non-dominated sorting of the population prior to selection; however, since each sub-GA explores within the search space of an individual variant and selects on the basis of that sub-fitness value, the search quality enhances significantly and each lower-level GA can carry over features of high-performing subsets of the full product family chromosome to the offspring population. While searching for high quality individual variants in lower levels, the upper level selects the next generation members with respect to the overall family objectives by applying nondominated sorting with respect to the commonality level and the overall performance value (formed by summing the sub-performances calculated at lower levels). Therefore, although the search for optimal platform configuration and variant design are performed in separate GAs, the aforementioned replacement scheme applied in each generation in the upper level differentiates the proposed method from multi-stage methods that find the optimal platform and variant design in separate stages. Finally, due to the parallel nature of this decomposed method, each sub-GA can be executed on a separate processor using the MPI library (Pacheco 1997) for sending and receiving data among nodes during the evolution.[8] Hence, in addition to the improved performance due to decomposition, it is possible to achieve further reductions in computational time using parallel processing.

5 Demonstration: universal electric motor family

The universal electric motor family example was first created by Simpson et al. (1999, 2001) and has since been applied as a case study in the product family optimization literature. In this example, the goal is to design a scaled-based product family of universal electric motors that satisfy a variety of torque requirements using common platforms. Hence, the optimization process involves selecting the platform and scaled design variables and their corresponding values so that the range of torque requirements is satisfied while the commonality among the motors is maximized, individual motor weight is minimized, and individual motor efficiency is maximized. The detailed analysis including all equations relating the motor design variables to the output parameters can be found in Simpson et al. (1999, 2001). Based on this formulation, the design of a single motor involves eight design variables (see Table 1), two equality and four inequality constraints (see Table 2), treating mass minimization and power maximization as the two objectives. However, as opposed to other physical components which can be shared for reducing tooling cost, any current value (within the initial range) could be drawn from the motor based on other motor

Table 2 Design constraints for the electric motor example

1. Torque (Nm) = {0.05, 0.1, 0.125, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40,0.5}
2. Power (W) = 300 (for all motors)
3. Feasible geometry for all motors: $t < r_o$
4. Maximum magnetizing intensity for each motor: M $\leq$ 5000 *Amp×turns/m*
5. Maximum mass of the each motor: $Mass \leq 2\ kg$
6. Minimum efficiency of each motor: $\eta \geq 15\%$

[8]It should be noted that the parallelization method applied in this paper is the direct benefit of the proposed decomposition scheme and should not be confused with general types of parallel GAs (e.g. fine-grain, coarse-grain and master-slave models), which are derived from the evolutionary nature of the GA and are independent of the specific problem being solved. Generic parallel GAs could additionally be used to solve subproblems in the proposed decomposition if the optimization of an individual product is too complex for a single GA or if further speedup is desired, but we do not pursue this possibility here.

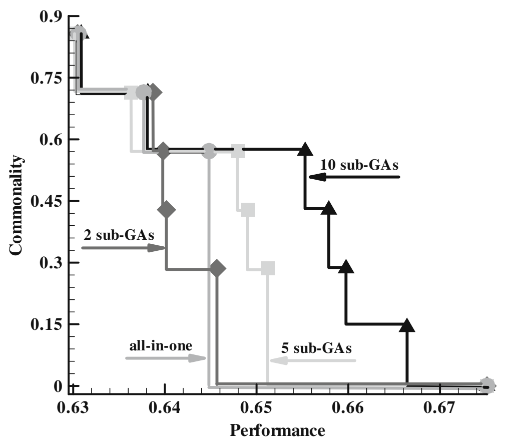


Fig. 8 Pareto fronts of the electric motor family for different decomposition schemes using all-or-none commonality

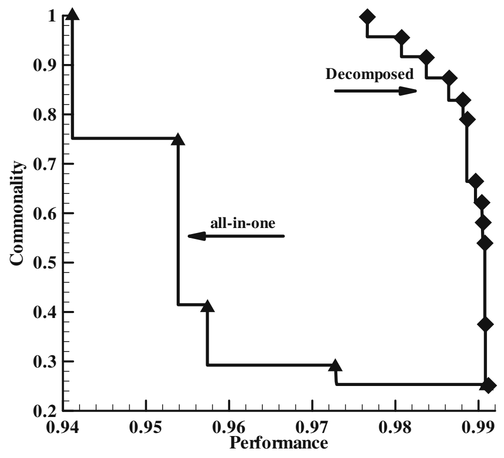


Fig. 9 Pareto fronts of GAA family for decomposed and all-in-one formulations using generalized commonality

characteristics and constraints. Hence, in this paper, current is written as a function of other variables using the power equality constraint[9] rather than being an independent variable and therefore is not considered for component sharing.

Hence, the number of design variables is reduced to 7 for each motor, and the power equality constraint is replaced by two inequality constraints representing the lower and upper bounds for current.

5.1 Product family objective functions

To demonstrate the tradeoff between commonality and individual performance as well as to visualize the Pareto curves conveniently; the three objectives introduced in Akundi et al. (2005) were reduced to two objectives by combining the two performance objectives into one using a fixed weighted sum:

$$f_1 = \sum_{i=1}^{10} w_1 \eta_i + w_2 \left(1 - m_i^*\right)$$
$$f_2 = \mathrm{CI} \tag{3}$$

in which η_i and m_i^* represent efficiency and the normalized mass (mass of each motor has been normalized by dividing it over the maximum allowable: m_{max} = 2 kg) for the i^{th} motor, respectively, and w_1 and w_2 are the weight coefficients, which are assumed to be equal ($w_1 = w_2 = 0.5$) in this paper. Moreover, all constraints are handled using the constrained dominated approach (Deb 2001).[10]

5.2 Decomposition

To compare the efficiency of the decomposed approach relative to the all-in-one formulation, the electric motor family was optimized for the restricted representation for commonality genes suggested by Simpson and D'Souza (2004) using several alternative decomposition schemes. First, the product family was decomposed into ten lower-level GAs, each optimizing a single product. A population size of 2,500 and maximum generation number of 800 were used. This parameter tuning was verified by running the same code using larger values for population size and maximum generation number, resulting in negligible improvement for the Pareto curves. Next, the same problem was solved using three alternative schemes: (1) five sub-GAs, each containing two motors; (2) two sub-GAs, each with five motors; and (3) the all-in-one formulation for all ten motors. The estimated Pareto curves are plotted in Fig. 8. Since we are interested in finding the benefit of decomposition in producing better results for the same computational cost, the same population size and maximum generation number (which implies the same number of function evaluations) was applied in all four cases.

[9]Power equality constraint is a 2nd order equation as a function of current and has two positive roots if any. Therefore, the roots are compared with respect to feasibility and objective value, and the better one is picked as the current value.

[10]Since GAs are generally inefficient for handling equality constraints directly and need a large population size for finding feasible solutions, the torque equality constraint has been implemented using an adaptive coefficient strategy in the constrained dominated approach.

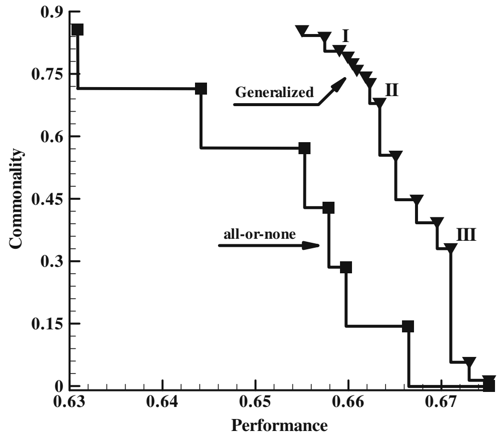


Fig. 10 Pareto fronts of the electric motor family for the generalized and all-or-none commonality definitions

Table 3 Platform configuration of points on the Pareto frontier in Fig. 10

Module	I	II	III
Nc	10	2, 4	2, 2
Ns	10	3, 7	4, 2, 2, 2
A_{wa}	10	9	5
A_{wf}	4, 4	4	4
r_o	5, 5	7	4
t	6	5, 2	4
L	10	2, 4	2, 2

As can be seen from Fig. 8, the all-in-one formulation cannot find a well-distributed optimal front: It produces only the high-commonality portion of the curve in that this region has a lower dimensionality. By increasing the number of sub-GAs (the extent of decomposition), the Pareto curve moves toward the optimal front, and best results are obtained for the most decomposed case.

While we focus here on results from the universal electric motor example, the trends observed in this case study are also consistent with results we obtained applying the proposed decomposition to the family of three general aviation aircraft (GAA) examined in Simpson

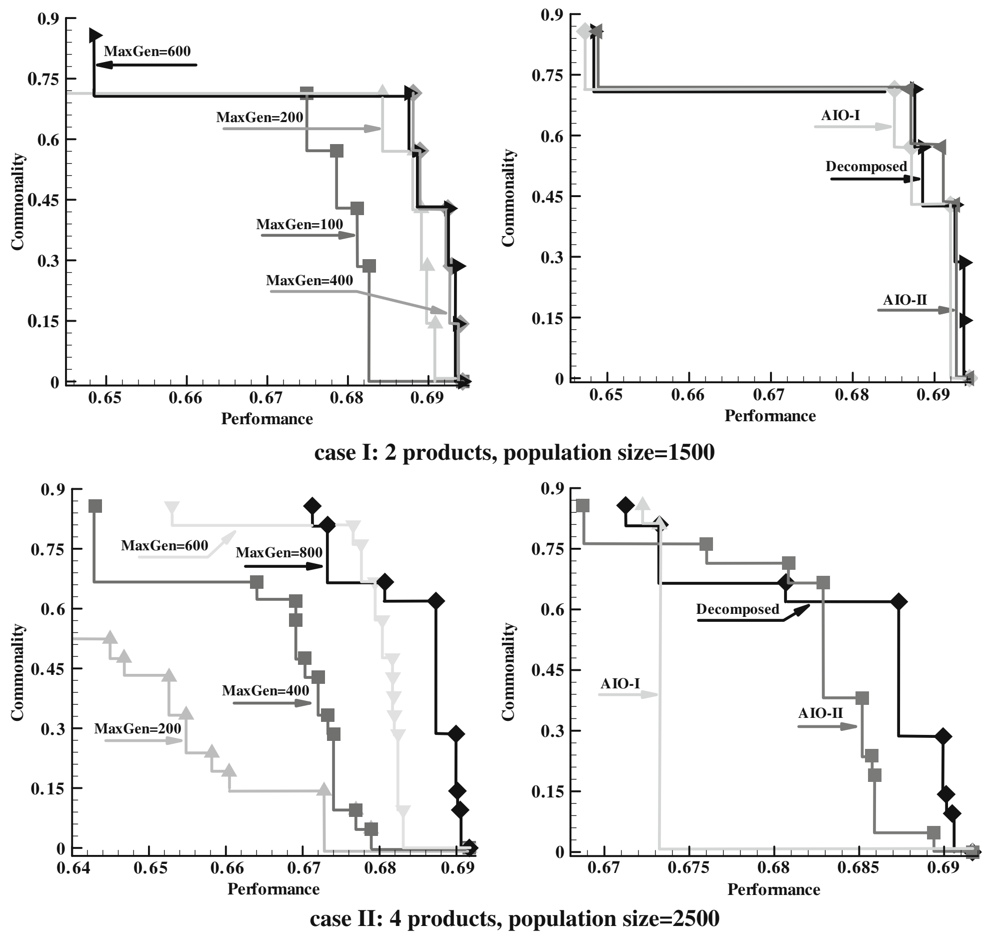
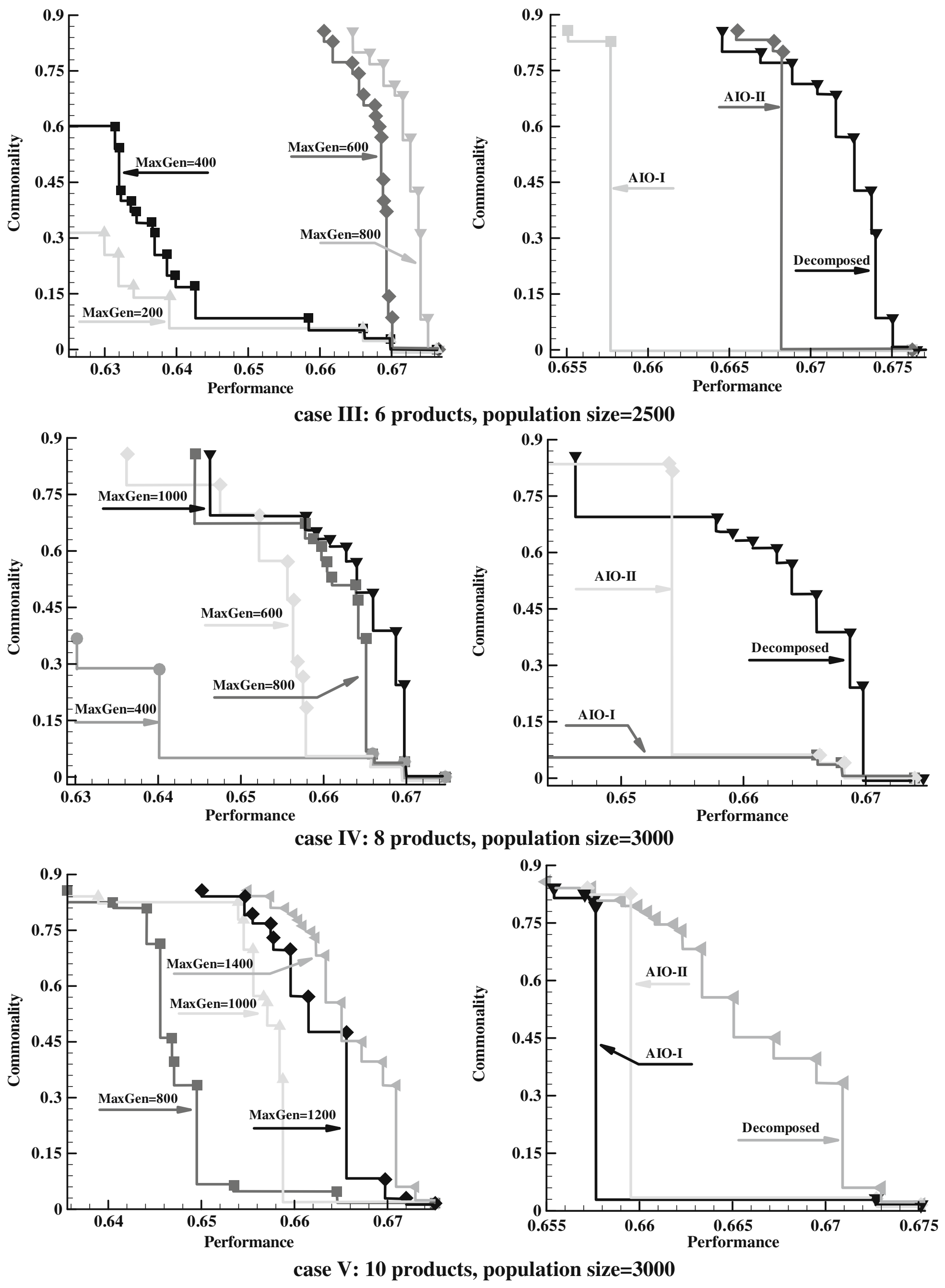


Fig. 11 Scalability of the proposed decomposition method versus the all-in-one problem

Fig. 11 (continued)



and D'Souza (2004). Specifically, the Pareto set obtained through decomposition is significantly superior to that obtained without decomposition (see Fig. 9) indicating the robustness of the proposed approach to solve families with different number of variants and components. Details of the GAA case study can be found in Khajavirad et al. (2007).

5.3 Generalization

As described in previous sections, using all-or-none commonality definition results in a loss of the benefit of component sharing among subsets of products, which is frequently applied in practice. The scalability benefit of decomposition is even more critical for addressing

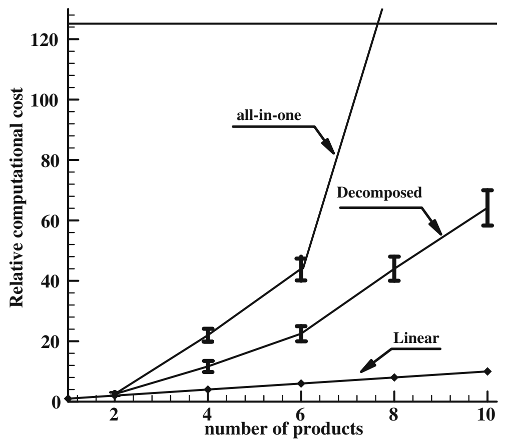


Fig. 12 Complexity of the proposed method as a function of the number of variants (Relative computational cost for *p* products = No. of function evaluations for *p* products/No. of function evaluations for optimizing a single product)

this generalized case: Since adding the two-dimensional chromosome along with the design variable chromosome and searching through all possible platform configurations adds to the complexity of the algorithm dramatically, the all-in-one formulation becomes impractical even for smaller numbers of products. For example, in the GAA case study with only three variants (Khajavirad et al. 2007), the all-in-one code failed to generate good results for the generalized case (see Fig. 9). This will be demonstrated numerically in section 5–5.

Therefore, in order to show the benefit of generalized commonality, the all-in-one formulation was decomposed to ten sub-GAs, each optimizing an individual variant. The population size and maximum generation number are 3,000 and 1,400, respectively, verified as before.

The Pareto frontier for the generalized commonality is depicted in Fig. 10 and compared with the all-or-none case using the same decomposition scheme. As can be seen from the figure, relaxing the all-or-none restriction to generalized commonality improves the product family performance dramatically and allows an average 30% of increased component sharing for the comparable level of performance leading to significant tooling cost savings.

In addition, to demonstrate the concept of generalized commonality, platform configurations for the three labeled points of the Pareto frontier are listed in Table 3. The numbers indicate the number of variants that share each component (1's are omitted). For instance, the notation {5, 2} indicates one design is shared among five variants while another design is common between two variants and the remaining three have distinct components. The presence of several sub-platforms for most of the components shows the importance of the generalization and also the efficiency of our new chromosome representation for capturing this feature.

5.4 Complexity of the decomposition scheme

To investigate the scalability of the proposed decomposition compared to the all-in-one problem; the electric motor example has been solved for different numbers of products. Pareto frontiers for 2, 4, 6, 8 and 10 variants are depicted in Fig. 11. Two graphs are shown for each case: The left hand shows the evolution of the decomposed algorithm, and the right hand compares the decomposed algorithm to the all-in-one approach. Specifically, the left hand graphs show the estimated Pareto front for several different values of MaxGen,[11] the maximum number of generations. Convergence was assumed when the average performance improvement between two consecutive 200-generation steps was less than 1%. The right hand graphs compare the converged estimated Pareto front from the decomposed MOGA to two all-in-one cases: (1) AIO-I represents the results of running the all-in-one algorithm for an equal number of function evaluations, and (2) AIO-II represents the results of running the all-in-one algorithm until it is within 1% of the decomposed solution or has exceeded the maximum allowed function evaluations,[12] whichever is less. Meanwhile, Fig. 12 compares the computational cost required to achieve convergence for both the decomposed and the all-in-one algorithms. Error bars represent the 200-generation step within which the 1% convergence criteria was achieved.

As can be observed from Figs. 11 and 12, as the number of variants in the family increases, the computational requirements increase exponentially for both methods. In the full decomposition scheme, increasing the number of variants has no effect on the size of each sub-GA; however, the 2D commonality chromosome size does grow, acting as the limiting factor for the algorithm scalability. This adverse effect could be seen from Fig. 12 by comparing the decomposed curve with the linear case. However, the relative benefit of

[11]Because of dynamic penalty function parameters for constraint handling that depend on the MaxGen parameter, the algorithm was restarted in each case from the same starting point.

[12]We set the maximum allowed number of function evaluations to twice the number of function evaluations required for solving the 10 product case using the decomposed algorithm.

Table 4 Parallelization times for the motor example using various decomposition schemes

Decomposition scheme	Time per generation (s)		
	Computation time	Communication and idle time	Total execution time
2-sub GAs (3 processors)	0.178	1.84	2.081
5-sub GAs (6 processors)	0.081	3.38	3.461
10-sub-GAs (11 processors)	0.045	4.39	4.435

decomposition over the all-in-one approach increases substantially as the number of variants increases, supporting improved scalability and the ability to solve larger problems than possible without decomposition.

5.5 Parallelization

As was shown in previous sections, decomposing the initial "all-in-one" formulation enhances the search quality of each sub-GA and decreases the total computational cost dramatically. To further reduce the computational time, the decomposed approach can be parallelized by running each sub-GA on a separate processor and using the MPI library for exchanging data among different nodes. Total execution time of the parallel codes can be divided into three main parts: computation time, communication time (i.e., time for exchanging data among processors) and idle time (e.g., synchronization time). Hence, the final speedup depends on how these three factors change by increasing the number of processors, which is problem specific. In our decomposition scheme, the computational time for each sub-GA is inversely proportional to the number of sub-GAs. This is due to the fact that the number of genes in each sub-GA chromosome is inversely proportional to the total number of sub-GAs and the MOGA operators act on each gene separately. However, increasing the number of processors increases the communication and idle time. Hence, parallelization is beneficial for cases in which the computation time is the dominant part of the total execution time, which in the case of GAs is determined by the computational cost of the fitness calculation phase.

The computation, communication and total execution time for one generation of the motor example are listed in Table 4 for 2, 5 and 10 sub-GAs (3, 6 and 11 processors) respectively. The reported times are the maximum time value among all the sub-GAs, i.e. the communication time and idle time are combined. As can be seen from the table, since each electric motor analysis (i.e., fitness calculation) involves only a number of analytic equations, and as result a very low computational cost (1.7×10^{-6} s), total execution time is dominated by the communication time which is increased by increasing the number of processors. Hence, the total execution time is increased by increasing the number of processors. However, as can be found from Table 4, the computation time is reduced considerably by increasing the number of processors and the communication time is negligible for cases involving time consuming product-level simulations. Hence, parallelization is not recommended for the motor example; however, for more computationally intensive applications, a high speedup can be achieved through parallel processing.

6 Conclusions and future work

In this paper, we introduced a new single-stage approach for solving the joint product family optimization problem using a unique decomposed MOGA formulation with a generalized commonality chromosome. The augmented chromosome representation introduced by Simpson and D'Souza (2004) was generalized to address component sharing among subsets of products. In order to improve the scalability of the proposed approach, the all-in-one MOGA was decomposed into a novel two-level optimization problem in which the upper-level GA finds the optimal platform configuration while each lower-level GA optimizes a subset of products in the family. The proposed approach was demonstrated by optimizing a family of universal electric motors. First, the effect of decomposition degree on the quality of estimated optimal fronts under fixed computational cost was investigated by solving the restricted commonality case using different decomposition schemes. The most decomposed case outperformed other schemes and found a well-converged and well-distributed optimal curve with a relatively low computational cost. Next, the same example was solved using the generalized commonality definition for the most decomposed case. Results show that the generalized commonality representation improves the optimal points dramatically, dominates all solutions of the all-or-none algorithm and captures the tradeoff between commonality and performance more effectively. These trends are consistent with a second case study of a family of three general aviation aircraft, thus supporting generalizability of the empirical results. Finally, the complexity of the decomposition scheme in terms of number of variants was examined by solving families with different numbers of products; results show the proposed decomposition improves the scalability of the all-in-one problem significantly; extending the applicability of the optimization algorithm to families with more variants. More scalable algorithms could be explored by investigating alternatives to the proposed 2D commonality chromosome for representing the generalized component sharing.

In conclusion, common restrictions of commonality to degrees of all-or-none may significantly limit the quality of the resulting solutions and the ability to take full advantage of commonality options. Thus, we recommend that all-or-none restrictions be used only when the firm is truly uninterested in generalized commonality for reasons of logistics, etc. Secondly, the proposed decomposition is significantly more efficient than all-in-one approaches, and it is able to generate evenly-distributed Pareto curves. We see no disadvantage to the decomposed approach, and we recommend the approach for future work in product line and product family optimization. In addition, for cases in which the fitness calculation phase (i.e., the product level simulation) involves high computational cost; a high speedup can be achieved by running the decomposed approach on a parallel machine.

For future work, we intend to examine deterministic global optimization approaches to solve the joint product family problem (Khajavirad and Michalek 2008b) and compare the true optimal front with those found using popular heuristics and local solvers in the product family optimization literature, quantifying the benefits and limitations of each group. In addition, current objectives for commonality and deviation from exogenous performance targets used in this paper, which are the standard in product family optimization, are somewhat artificial and limited substitutes for the benefits of commonality and differentiation. Future work aims to introduce methods for quantifying cost benefits of commonality and revenue benefits of differentiation in a heterogeneous marketplace in order to make the most profitable tradeoff in product family planning and design (Kumar et al. 2006; Michalek et al. 2006b).

Acknowledgements This work is supported in part by the Pennsylvania Infrastructure Technology Alliance, a partnership of Carnegie Mellon, Lehigh University, and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DCED). Dr. Simpson also acknowledges support from the National Science Foundation under CAREER Award No. DMI-0133923. Any opinions, findings, and conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

References

Akundi S, Simpson TW, Reed PM (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

de Weck O (2005) Determining product platform extent. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design: methods and applications. Springer, New York, pp 241–301

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, West Sussex, UK

Deb K, Agrawal S, Pratap A, 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

Fujita K (2002) Product variety optimization under modular architecture. Comput Aided Des 34(12):953–965

Fujita K, Yoshida H (2001) Product variety optimization: simultaneous optimization of module combination and module attributes. ASME design engineering technical conferences—design automation conference. Pittsburgh, PA

Hassan R, de Weck O, Springmann P (2004) Architecting a communication satellite product line. In: 22nd AIAA international communications satellite systems conference & exhibit (ICSSC). Monterey, CA

Khajavirad A, Michalek JJ (2008a) A decomposed approach for solving the joint product family platform selection and design. Mech Des v130 p071101

Khajavirad A, Michalek JJ (2008b) A deterministic Lagrangian-based global optimization approach for large scale decomposable problems. In: ASME international design engineering technical conferences & computers and information in engineering conference IDETC/CIE. ASME, New York, USA

Khajavirad A, Michalek JJ, Simpson TW (2007) A decomposed genetic algorithm for solving the joint product family optimization problem. In: 3rd AIAA multidisciplinary design optimization specialists conference. Honolulu, HI

Khire RA, Messac A, Simpson TW (2006) Optimal design of product families using Selection-Integrated Optimization (SIO) Methodology. In: 11th AIAA/ISSMO symposium on multidisciplinary analysis and optimization. Portsmouth, VA

Kokkolaras M, Fellini R, Kim HM, Michelena NF, Papalambros PY (2002) Extension of the target cascading formulation to the design of product families. Struct Multidisc Optim 24(4):293–301

Kumar D, Chen W, Simpson TW (2006) A market-driven approach to the design of platform-based product families. In: AIAA/ISSMO multidisciplinary analysis and optimization conference. Portsmouth, VA

Martin M, Ishii K (1996) Design for variety: a methodology for understanding the costs of product proliferation. In: Design theory and methodology—DTM'96. Irvine, CA

Messac A, Martinez MP, Simpson TW (2002) Effective product family design using physical programming. Eng Optim 34(3):245–261

Michalek JJ, Ceryan O, Papalambros PY, Koren Y (2006a) Balancing marketing and manufacturing objectives in product line design. J Mech Des 128(6):1196–1204

Michalek JJ, Ceryan O, Papalambros PY, Koren Y (2006b) Balancing marketing and manufacturing objectives in product line design. ASME J Mech Des 128(6):1196–1204

Pacheco P (1997) Parallel programming with MPI. Morgan Kaufmann, San Francisco, CA

Simpson TW (2005) Methods for optimizing product platforms and product families: overview and classification. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design: methods and applications. Springer, New York, pp 133–156

Simpson TW, D'Souza BS (2004) Assessing variable levels of platform commonality within a product family using a multiobjective genetic algorithm. Concurr Eng Res Appl 12(2):119–129

Simpson TW, Maier JRA, Mistree F (1999) A product platform concept exploration method for product family design. In: Design theory and methodology—DTM'99. Las Vegas, Nevada

Simpson TW, Maier JRA, Mistree F (2001) Product platform design: method and application. Res Eng Des 13(1):2–22

Thevenot HJ, Simpson TW (2006) Commonality indices for product family design: a detailed comparison. J Eng Des 17(2):99–119