INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED03 STOCKHOLM, AUGUST 19-21, 2003

A RIGOROUS FRAMEWORK FOR MAKING COMMONALITY AND MODULARITY DECISIONS IN OPTIMAL DESIGN OF PRODUCT FAMILIES

Ryan Fellini, Michael Kokkolaras, and Panos Y. Papalambros

Abstract

This article proposes a set of math-based approaches for making commonality and modularity decisions when designing product families. The performance of products that share some components is usually compromised relative to the individual optimum. This deviation occurs because of the commonality constraints that are included in the optimal design problem, especially when the objectives in the multicriteria formulation are conflicting. Choosing components for sharing may depend on what performance deviations can be tolerated. We present rigorous strategies for choosing components to be shared without exceeding user-specified bounds on performance. In addition, we consider modularity to be an outcome of the commonality decision process.

Keywords: Optimal Design, Product Families, Commonality, Modularity

1 Introduction

We refer to products that have similar architecture but different functional requirements as variants. Product variants can be derived based on a platform, i.e., a set of common parts, in which case they form a product family. Product platforms enable the development of variants for rapid adjustment to changing market needs while keeping development costs and time-cycles low [1,2]. The functional requirements of product variants may be conflicting, in which case family optimal designs are compromised relative to individually optimized designs [3]. The design challenge is to maximize commonality while minimizing individual performance losses.

We focus on math-based methodologies for determining which components should be shared. Chen *et al*. and Nayak *et al*. use robust design principles to select the platform for a family of scalable products [4,5]. Gonzalez-Zugasti *et al*. and Gonzalez-Zugasti and Otto cluster parts into modules to reduce the problem size, and then solve the combinatorial design problem for modular product families [6,7]. Fujita and Yoshida follow a similar approach for simultaneous optimization of module combination and attributes [8]. The latter is based on previous work of Fujita *et al*. [9,10,11]. A genetic algorithm (GA) is linked with sequential quadratic programming (SQP) in the above methods. The GA is used to choose the modules to be shared and SQP is used to solve the family design problem. D'Souza, B. and Simpson have also adopted GAs for solving the commonality selection and family design problems [12]. The number of possible combinations increases exponentially the number of products and/or variables. Therefore, combinatorial algorithms like GAs may be insufficient for solving even problems of modest size.

In this article we integrate two approaches developed in previous work into a rigorous framework of commonality strategies. The first approach uses first-order information, obtained from solving the individual optimal design problems, to compute a metric for performance deviations attributed to component sharing [13]. The second approach uses a relaxed formulation of the combinatorial problem to maximize commonality, while satisfying designer-specified bounds on individual performance losses [14]. The main idea is to use the first approach as a filter to reduce the platform selection problem size and the second approach to maximize commonality while minimizing individual performance losses. However, there are more than one ways to combine the two approaches. We propose some here.

The article is organized as follows: after presenting some necessary definitions to establish a common glossary, we review the two approaches. We then formulate the general combined methodology, and discuss different options. We conclude with a discussion on how modularity information can be extracted from the commonality decision process.

2 Definitions

A product is an artifact created with the intent to serve user needs, i.e., to satisfy some functional requirements. In the present context the most fundamental term used when discussing product design is the term model. A model is a mathematical representation of the artifact and is a core element of engineering design and analysis. A model accepts a set of variables as inputs and returns a set of responses as outputs. A collection of products will be represented by the set $P = \{p_1, p_2, \ldots\}$ used to distinguish each product $p \in P$. A product p is associated with a vector of input variables $\mathbf{x}^p \in \mathbb{R}^{n^p}$ and a vector of output responses $\mathbf{R}^p \in \mathbb{R}^{d^p}$. In the context of a model the vector $\mathbf{x}^p$ is assumed to describe a given design fully, namely, if values are assigned to all the components of $\mathbf{x}^p$ then p is defined uniquely. Similarly, $\mathbf{R}^p$ is assumed to represent all responses of interest in evaluating the attributes of a given design. Clearly, these are strong assumptions but necessary in the use of any analytical tool. The designer is expected to augment the insights gained from analysis to reach the best design possible by including non-analytical considerations.

Assuming that we can map the functional requirements of a product to model responses, we formulate the optimal design model for a product p as

$$\begin{aligned} \max_{\mathbf{x}^p} \quad & f^p(\mathbf{x}^p) \\ \text{subject to} \quad & \mathbf{g}^p(\mathbf{x}^p) \le \mathbf{0} \\ & \mathbf{h}^p(\mathbf{x}^p) = \mathbf{0}, \end{aligned} \tag{1}$$

where the objective function f^p, representing performance, is maximized, and additional product requirements are represented by design inequality and equality constraint functions $\mathbf{g}^p$ and $\mathbf{h}^p$, respectively.

A component is defined as a manufactured object that is the smallest (indivisible) element of a product. An assumption is made that the product can readily be decomposed into its base components. A component will have its own set of design variables, $\mathbf{x}^{c^p} \subset \mathbf{x}^p$, which are a subset of the product design variables. An assembly is an object constructed from a set of connected components.

It should be noted that the point of reference when defining a product can easily change. The engine of a vehicle is an assembly, which is part of an automobile, but could be designed as a product composed of various assemblies and components such as the exhaust system, fuel injection system, etc. In this case the point of reference changes and the engine is at the "top level" of the product hierarchical description.

The product architecture is the configuration of components and assemblies within a product and includes the set of instructions for assembly of the product. The costs associated with manufacturing are not considered in this work. Variable costs having to do with mass production, etc., are also ignored.

A product platform is the set of all elements, interfaces, manufacturing and systems processes that are common in a set of products. This article focuses on commonality in product design only, therefore it is limited to parts sharing only. The following notation is used to describe a product platform. The set S^{pq} consists of the index pairs of elements that are shared between two products p and q. The set $S = \{S^{pq} \mid p, q \in P; p < q\}$ describes element sharing throughout the family. The empty set $S_{\varnothing}$ defines the null platform (no sharing between products).

Various types of sharing can be used based on the definitions in the previous section. In component sharing one or more components are common across a family of products. Likewise, assemblies can be shared between products. In addition, it is possible to share "scaled" versions of components. Mathematically this can be described as variable sharing, where components are based on a platform (of variables) themselves.

A product family is the set of products that share product platform. The set S maps the relationships between products in P. The product family can have an objective f^F which is a function of the design variables of the entire family, $\mathbf{x}^F = \left[\mathbf{x}^{p_1}, \mathbf{x}^{p_2}, \ldots\right]$. Such a family objective could be, e.g., cost or profit. Likewise, we can have family constraints $\mathbf{g}^F$ and $\mathbf{h}^F$, e.g., for a family of automobiles there may be constraints on production capacity and/or corporate average fuel economy (CAFE) penalties. Assuming that the commonality decisions have been made, the multiobjective optimal family design problem is formulated as

$$\begin{aligned} \max_{\mathbf{x}^F} \quad & \{f^F, f^p(\mathbf{x}^p)\} \qquad \forall p, q \in P, (i,j) \in S^{pq}, p < q \\ \text{subject to} \quad & \mathbf{g}^F(\mathbf{x}^F) \le \mathbf{0} \; ; \; \mathbf{g}^p(\mathbf{x}^p) \le \mathbf{0} \\ & \mathbf{h}^F(\mathbf{x}^F) = \mathbf{0} \; ; \; \mathbf{h}^p(\mathbf{x}^p) = \mathbf{0} \\ & x_i^p = x_j^q \end{aligned} \tag{2}$$

A module is a special case of a component in a product that produces variety when interchanged (*cf.* Figure 1).

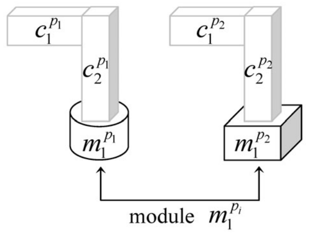


Figure 1. Modular product with one interchangeable module.

Ideally a module should allow for the capability to modify the characteristics defining the product independently. For example, one factor in choosing a computer is sound capability. Therefore as a module the sound card can be interchanged to control this specific metric and does not influence other metrics such as the video quality or the processing speed. The assumption will also be made that a module need not be connected. For example, two components in a product that influence a particular product response, R_i^p, may be not connected, but can be exchanged simultaneously to create a new product. The set $M^p = \{\mathrm{m}_1^{\mathrm{p}}, \mathrm{m}_2^{\mathrm{p}}, \ldots\}$ is used to represent the modules in a product p. A module is formed from a subset of the components which make up a product, and we assume that components are not shared between modules. Again, this is an important assumption. $\mathbf{x}^{m^p}$ consists of all the component design variables included in the module.

Two important aspects regarding modules is that they have an interface defined for interchangeability and that the interactions between the module and its surrounding should be minimized to afford better encapsulation of its design and function. For example, if the engine block is considered a module within a set of engine products and the piston is part of the platform, the bore of the engine block modules should fit that of the piston platform. This common interface is the platform shared amongst modules. Modules themselves can be shared by a subset of the products, forming a modular (sub)platform (*cf.* Figure 2).

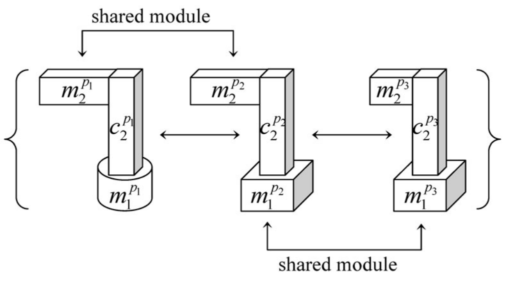


Figure 2. Modular products with module sharing among a subset of products.

An example of a modular platform can be a family of automobiles where variety is produced by interchanging engines. While two of the vehicles may use the same engine, a third vehicle may use a different engine (where the engines are interchangeable among the three vehicles). The first two vehicles would be differentiated perhaps by a second module influencing some other vehicle metric.

3 Platform Selection

In this section we review two approaches developed for solving the commonality selection problem.

3.1 Performance Deviation Vector

In this approach we use optimality and sensitivity information obtained from solving the individual optimal design problems to estimate potential deviation from the null-platform optimal design incurred by sharing parts with other product variants. Specifically, assuming that individual optima lie "close" enough to each other and that the family design will be in their convex hull, we use a first-order Taylor series approximation to derive an upper bound

Π_i on the performance deviation due to sharing variable x_i [13]. For two variants A and B this upper bound is given by

$$\Pi_i = (1-\lambda_i)\left(\left|\nabla_i f^{A,o}\right|\delta_i + \sum_{j\in G^A}\max\left(\nabla_i g_j^{A,o}\delta_i, 0\right)\right) + \lambda_i\left(\left|\nabla_i f^{B,o}\right|\delta_i + \sum_{j\in G^B}\max\left(\nabla_i g_j^{B,o}\delta_i, 0\right)\right), \quad (3)$$

where $0 \le \lambda_i \le 1$, $\delta_i = \left|x_i^{A,o} - x_i^{B,o}\right|$, the superscript ^{o} denotes null-platform, and G^p is the set of active constraints of product *p*. The designer can sort the values $\Pi_i\ \forall i$ in a so-called performance deviation vector (PDV) and base his decision on sharing according to some threshold. An example on the possible use of the PDV is illustrated in Figure 6 [13]. In this example, there is a total of 63 variables that can be shared. The first 25 variables have zero performance deviation, i.e., the optimal design values corresponding to the null-platform optima were identical. Therefore, these variables correspond to "naturally" shared components. The designer may want to share nineteen additional variables since the performance deviation value is below 0.01 for these variables. In this case, the designer chooses according to an absolute threshold. An alternative is to choose according to a relative increase of performance deviation, i.e., using the ratio $(\Pi_{i+1} - \Pi_i)/\Pi_i$.

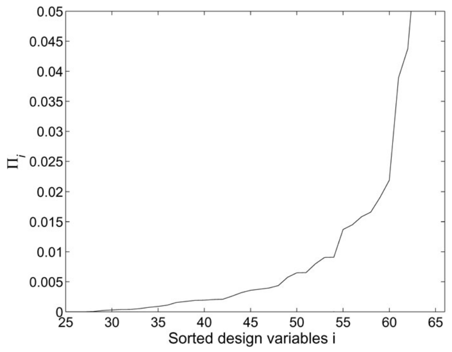


Figure 6. Graphical representation of the sorted PDV.

The performance deviation vector approach is computationally quite affordable but relies on the above mentioned assumptions and involves the heuristic of selecting a threshold for making commonality decisions. It is thus more appropriate for what we refer to as "mild" variants.

3.2 Relaxed Combinatorial Problem

This approach enables the designer to choose what performance losses are acceptable relative to null-platform optima. Component sharing is then determined through the solution of a relaxed combinatorial commonality maximization combinatorial problem subject to these performance bounds. The original mixed-discrete optimization problem has been reformulated using a continuous function to approximate the binary decisions [14]:

$$\begin{aligned} \min_{\mathbf{x}=[\mathbf{x}^{p1},\mathbf{x}^{p2},\ldots]} \quad & \sum_{(i,j)pq} D_\alpha\left(x_i^p - x_j^q\right) \quad \forall p,q \in P, (i,j)\in S^{pq}, p<q \\ \text{subject to} \quad & \mathbf{g}^p(\mathbf{x}^p) \le \mathbf{0} \\ & \mathbf{h}^p(\mathbf{x}^p) = \mathbf{0} \\ & f^p(\mathbf{x}^p) \ge (1-L^p)\, f^{p,o}, \end{aligned} \quad (4)$$

where $D_\alpha\left(x_i^p - x_j^q\right) = 1 - 1/\left(((x_i^p - x_j^q)/\alpha)^2 + 1\right)$. In our experience, a good value for α is 0.05. The designer can solve Problem (3) for different loss bounds L^p until he/she is satisfied with the tradeoff between maximizing commonality and minimizing individual performance losses. The relaxed combinatorial approach is free of simplifying assumptions but is computationally expensive, especially as the number of product variants and sharing candidates increases.

4 Combined Commonality Strategies

In this section we propose a number of available options for combining the two approaches to take advantage of their strengths. The first approach has the advantage of being very inexpensive and allows for a reasonable approximate ranking of the variables "shareability". The second approach is quite accurate in that one can specify a bound on performance deviation and then optimize to maximize commonality with respect to this bound. One modification to the first-order method must be made to facilitate the integration of the two approaches. This modification is necessary for commonality decisions involving components specified by vectors of design variables. This is accomplished by aggregating the deviation values of the design variables defining a component c into a single performance deviation value Π_c.

4.1 Linking the first-order method with an optimizer

The first option is to link the performance deviation vector approach to an optimization algorithm. This is based on the assumption that the components have been sorted "correctly" with respect to their influence to the product design. The algorithm can be as simple as the bisection or golden section methods or one may implement more sophisticated derivative-free algorithms. Gradient-based algorithms should only be used with caution because of the noisy nature of the vector. The idea is as follows. We can compute the maximum loss on performance by finding the so-called total-platform designs by solving the family design problem - Problem (2) - assuming that all design variables are shared. Since our objective is to maximize component sharing, we want to move to the right on the horizontal axis. At each iteration we optimize the family designs given the selected level of sharing by solving Problem (2), and determine our actual performance deviation. If performance loss is acceptable we select a bigger platform, otherwise a smaller one. Convergence is achieved when the desired bound on performance loss is met given maximum commonality. Figure 7 illustrates this approach for three iterations of the bisection method.

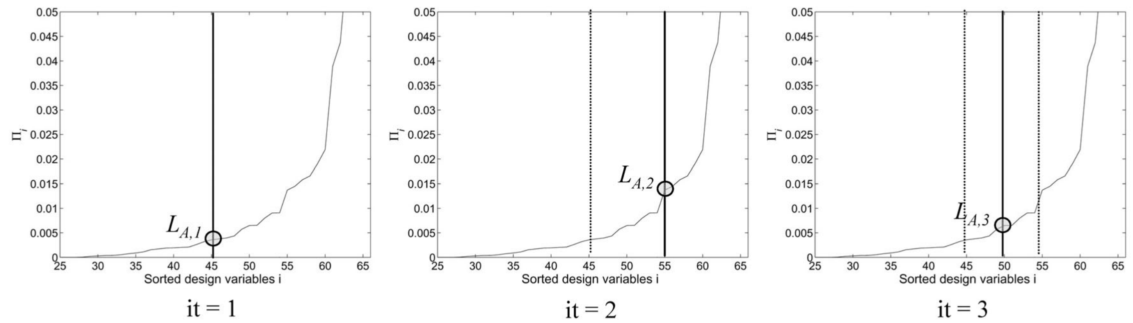


Figure 7. Optimizing over the performance deviation vector.

4.2 Using the first-order method as a filter

Another straightforward approach to combining the approaches is to use the first-order method as a filter to reduce the problem size solved by the relaxed combinatorial problem. We will review two alternative ways to do this.

The first alternative uses the first-order method to assume an initial set of components that can be shared. This is illustrated on the left plot of Figure 8.

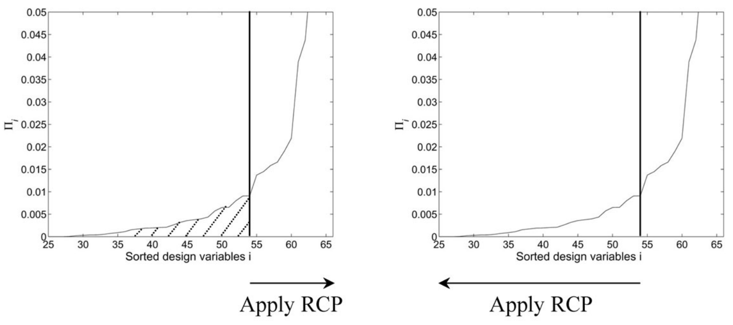


Figure 8. Using the performance deviation vector to reduce the relaxed combinatorial problem size.

The designer decides to share a number of components by means of a threshold according to the discussion in Section 3.1. Solving the relaxed combinatorial problem on the remaining candidates, one can determine if more components can be shared.

Alternatively, one can use the first-order method to determine the candidate platform and solve the relaxed combinatorial problem to determine if, or how many, of the components in the candidate platform can be actually shared. This alternative is illustrated on the right plot of Figure 8.

4.3 Combining the approaches into an iterative method

The previous options can also be combined into an iterative method. The motivation is to formulate a strategy that uses the performance deviation vector information to determine the smallest problem size possible for the relaxed combinatorial problem. In the example of Figure 9, a threshold of 0.01 we observe that the size of the relaxed combinatorial problem is smaller when considering the remaining candidates. If we happen to determine that we cannot share any additional components, we may want to reduce the threshold and solve the relaxed combinatorial problem on a subset of the initial candidate platform, and so on.

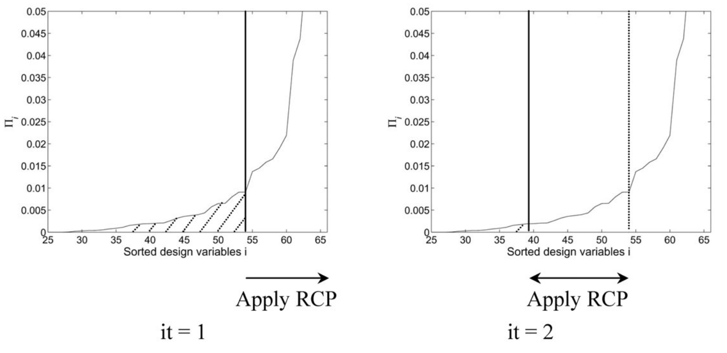
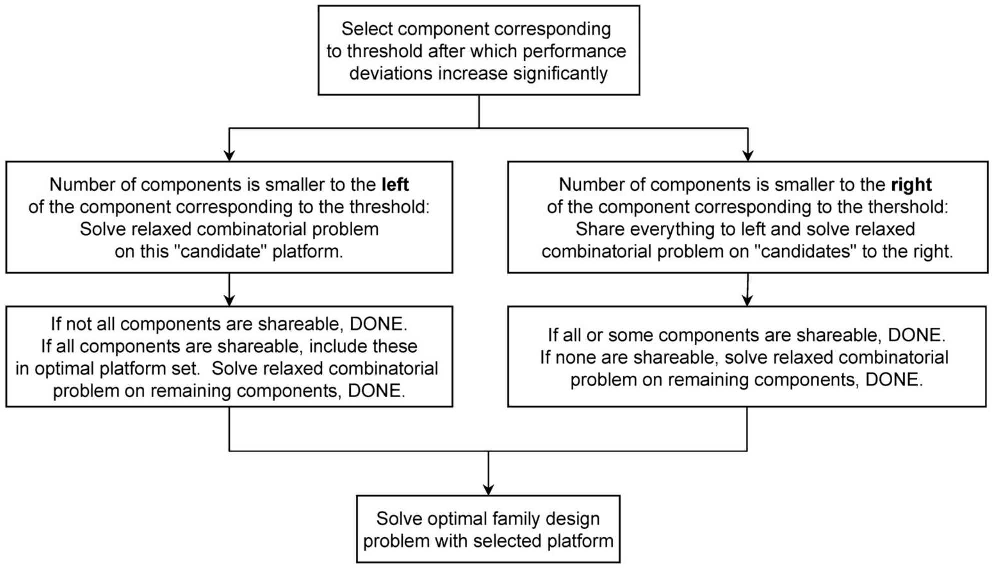


Figure 9. Iterative solution of the relaxed combinatorial problem.

This proposed iterative method is summarized in the following diagram:


Select component corresponding to threshold after which performance deviations increase significantly
Number of components is smaller to the **left** of the component corresponding to the threshold: Solve relaxed combinatorial problem on this "candidate" platform.
Number of components is smaller to the **right** of the component corresponding to the thershold: Share everything to left and solve relaxed combinatorial problem on "candidates" to the right.
If not all components are shareable, DONE. If all components are shareable, include these in optimal platform set. Solve relaxed combinatorial problem on remaining components, DONE.
If all or some components are shareable, DONE. If none are shareable, solve relaxed combinatorial problem on remaining components, DONE.
Solve optimal family design problem with selected platform

5 Modularity Decisions

After making commonality decisions by means of one of the alternatives described in the previous section we are left with a set of components which provide the variety in the family. The variants are derived by changing these components. Therefore, components that are part of the module set are readily identified as the ones that are not being shared by all products. Subplatforms (sharing between a subset of products) represent in effect the sharing of a particular module. Note that we can also consider the platform itself as a module, namely, a module that produces no variety.

The final step is to cluster components that are now members of the module set M into individual modules m_i^p. This can be accomplished by using partitioning algorithms [15] to cluster components that impact various critical responses of the product (*cf.* Figure 10).

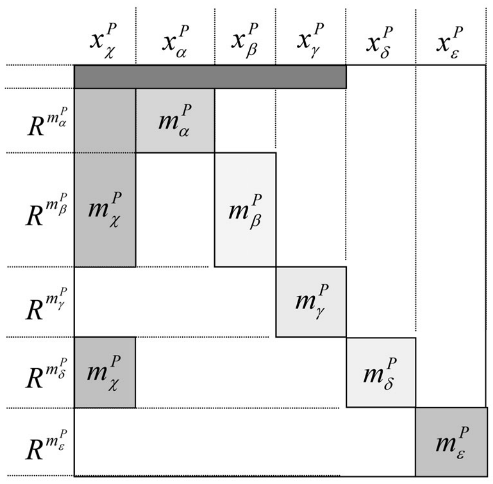


Figure 10. Extracting modularity from partitioned dependency table.

These algorithms can be applied to the functional dependency table (FDT) of the modules m_i^p. This partitioning also aids the designer in understanding the interactions in order to choose modules that can be designed and interchanged independently.

6 Conclusions

A rigorous framework for making commonality decisions when designing a product family has been developed by combining two previously developed math-based approaches. Several combination alternatives have been presented to provide more than one option to the designer, who can utilize them on a case-by-case basis. It has been also demonstrated that modularity can be extracted as a byproduct of the commonality decision process. Some of the proposed strategies of this article have been applied successfully to automotive body structure and engine family design problems [16,17].

7 Acknowledgement

This research was partially supported by the Automotive Research Center, a US Army Center of Excellence in Modeling and Simulation of Ground Vehicles, by a US Army Dual-Use Science and Technology Project, and by the General Motors Collaborative Research Laboratory at the University of Michigan. This support is gratefully acknowledged.

References

[1] Meyer, M. and Lehnerd, A., "The Power of Product Platforms", The Free Press, New York, 1997.

[2] Ericsson, A. and Erixon, G., "Controlling Design Variants", ASME Press, New York, 1999.

[3] Nelson, S., Parkinson, M., and Papalambros, P.Y., "Multicriteria optimization in product platform design", in Proceedings of the 25th ASME Design Engineering Technical Conferences, Las Vegas, Nevada, 1999 paper no. DAC-8676, also appeared in ASME Journal of Mechanical Design, 123(2):199-204, June, 2001.

[4] Chen, W., Allen, J., and Mistree, F., "The robust concept exploration method for enhancing concurrent systems design", Journal of Concurrent Engineering: Research and Applications, 5(3), 203–217, 1997.

[5] Nayak, R., Chen, W., and Simpson, T., "A variation-based methodology for product family design", in Proceedings of the 26th ASME Design Engineering Technical Conferences, Baltimore, Maryland, 2000, paper no. DAC-14264, also appeared in Engineering Optimization, 34(1):69-81, 2002.

[6] Gonzalez-Zugasti, J., Otto, K., and Baker, J., "A method for architecting product platforms with an application to interplanetary mission design", in Proceedings of the 24th ASME Design Engineering Technical Conferences, Atlanta, Georgia, 1998, paper no. DAC-5608.

[7] Gonzalez-Zugasti, J. and Otto, K., "Modular platform-based product family design", in Proceedings of the 26th ASME Design Engineering Technical Conferences, Baltimore, Maryland, 2000, paper no. DAC-14238.

[8] Fujita, K. and Yoshida, H., "Product variety optimization: Simultaneous optimization of module combination and module attributes", in Proceedings of the 27th ASME Design Engineering Technical Conferences, Pittsburgh, Pensylvania, 2001, paper no. DAC-21058.

[9] Fujita, K., Akagi, S., Yoneda, T., and Ishikawa, M., "Simultaneous optimization of product family sharing system structure and configuration", in Proceedings of the 24th ASME Design Engineering Technical Conferences, Atlanta, Georgia, 1998, paper no. DFM-5722.

[10] Fujita, K., Sakaguchi, H., and Akagi, S., "Product variety deployment and its optimization under modular architecture and module commonalization", in Proceedings of the 25th ASME Design Engineering Technical Conferences, Las Vegas, Nevada, 1999, paper no. DFM-8923.

[11] Fujita, K., "Product variety optimization under modular architecture", in Proceedings of the Third International Symposium on Tools and Methods of Competitive Engineering, 2000, Delft, The Netherlands.

[12] D'Souza, B. and Simpson, T., "A genetic algorithm based method for product familydesign optimization", in Proceedings of the 28th ASME Design Engineering Technical Conferences, Montreal, Quebec, Canada, 2002, paper no. DAC-34106.

[13] Fellini, R., Kokkolaras, M., Michelena, N., Papalambros, P.Y., Saitou, K., Perez-Duarte, A., and Fenyes, P., "A sensitivity-based commonality strategy for family products of mild variation, with application to automotive body structures", in Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, 2002, paper no. AIAA-2002-5610.

[14] Fellini, R., Kokkolaras, M., Papalambros, P.Y., and Perez-Duarte, A., "Platform selection under performance loss constraints in optimal design of product families", in Proceedings of the 28th ASME Design Engineering Technical Conferences, Montreal, Quebec, Canada, 2002, paper no. DAC-34099.

[15] Michelena, N. and Papalambros, PY., "A hypergraph framework for optimal model-based decomposition of design problems", Journal of Computational Optimization and Applications, Vol. 8, No. 2, pp. 173-196, September, 1997.

[16] Fellini, R., Kokkolaras, M., and Papalambros, P.Y., "Quantitative Platform Selection in Optimal Design of Product Families, with Application to Automotive Engine Design", submitted to Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, January 2003.

[17] Fellini, R., "A Model-Based Methodology for Product Family Design", PhD thesis, University of Michigan, 2003.

For more information please contact:

Ryan Fellini, University of Michigan, Department of Mechanical Engineering,
2350 Hayward St., Ann Arbor, Michigan, 48109-2125, United States
Tel: +1 734 647-8401, Fax: +1 734 647-8403, E-mail: rfellini@umich.edu
URL: http://ode.engin.umich.edu