Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2015), *29*, 333–350.
© Cambridge University Press 2015 0890-0604/15
doi:10.1017/S0890060415000372

A survey on automating configuration and parameterization in evolutionary design exploration

JULIAN R. EICHHOFF AND DIETER ROLLER
Institute of Computer-Aided Product Development Systems, University of Stuttgart, Stuttgart, Germany

(Received September 15, 2014; Accepted April 23, 2015)

Abstract

Configuration and parameterization of optimization frameworks for the computational support of design exploration can become an exclusive barrier for the adoption of such systems by engineers. This work addresses the problem of defining the elements that constitute a multiple-objective design optimization problem, that is, design variables, constants, objective functions, and constraint functions. In light of this, contributions are reviewed from the field of evolutionary design optimization with respect to their concrete implementation for design exploration. Machine learning and natural language processing are supposed to facilitate feasible approaches to the support of configuration and parameterization. Hence, the authors further review promising machine learning and natural language processing methods for automatic knowledge elicitation and formalization with respect to their implementation for evolutionary design optimization. These methods come from the fields of product attribute extraction, clustering of design solutions, relationship discovery, computation of objective functions, metamodeling, and design pattern extraction.

Keywords: Conceptual Design; Design Automation; Design Optimization; Evolutionary Computing; Knowledge Engineering; Machine Learning

1. INTRODUCTION

"The task of an engineer is to find solutions for technical problems" (Pahl et al., 2007). A multitude of such design tasks that affect different aspects of a technical product arise along the process of product development. The overall task that has to be accomplished by an engineering team can be paraphrased as search over a set of realizable technical artifacts for alternatives that show the highest compliance with a set of objectives. This directly lends a general scheme for representing design decisions in terms of mathematical optimization (Marler & Arora, 2004).

However, typically such problems are not fixed from the beginning. Multiple optimization problems evolve during design space exploration over consecutive design phases and reiterations. Consider the seminal work of Schon (1992), where he describes the act of exploring different design alternatives as a process of "seeing–moving–seeing." From an abstract perspective, formulating and solving an optimization problem can be seen as a *moving* step, whereas interpreting the results and making conclusions about the formulation of the next optimization problem is considered *seeing*.

Although computer-aided methods for solving optimization problems have proven very valuable over the last decades (Marler & Arora, 2004), most of these works solely address scattered and very specific *moving* steps. In order to automate design exploration over chains of *seeing–moving–seeing* and thus provide continuous support over several design phases, new approaches for formulating consecutive design problems are needed; that is, the *seeing* step has to be implemented.

Formulating an optimization problem usually requires design variables to be defined together with their ranges, constants (e.g., weights and other parameters), and functions over these elements used for setting objectives and constraints. In this paper, these tasks are paraphrased using the following notions of *parameterization* and *configuration*:

1. *Multiple-objective design optimization problem:* Given a vector of *design variables* $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, a vector of *constants* $\mathbf{c} = (c_1, c_2, \ldots, c_m)^T$, a vector of *objective functions* $\mathbf{f}(\mathbf{x}, \mathbf{c}) = [f_1(\mathbf{x}, \mathbf{c}), f_2(\mathbf{x}, \mathbf{c}), \ldots, f_k(\mathbf{x}, \mathbf{c})]^T$, a set of *inequality constraint functions* $g_i(\mathbf{x}, \mathbf{c})$ with $i = 1, 2, \ldots, p$, and a set of *equality constraint functions* $h_j(\mathbf{x}, \mathbf{c}) = 0$ with $j = 1, 2, \ldots, q$, then $\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{c})$

Reprint requests to: Julian Eichhoff, Institut für Rechnergestützte Ingenieursysteme, Universität Stuttgart, Universitätsstrasse 38, Stuttgart D-70569, Germany. E-mail: julian.eichhoff@informatik.uni-stuttgart.de

subject to $g_i(\boldsymbol{x}, \boldsymbol{c}) \leq 0$ and $h_j(\boldsymbol{x}, \boldsymbol{c}) = 0$ is called a *multi-objective optimization problem*.

2. *Parameterization:* The preparatory act of defining the value of each constant within $\boldsymbol{c}$ is called *parameterization*.
3. *Configuration:* The act of specifying the ranges for each design variable within $\boldsymbol{x}$ together with the definition of functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$, $g_i(\boldsymbol{x}, \boldsymbol{c})$, and $h_j(\boldsymbol{x}, \boldsymbol{c})$ is called *configuration*.

The present work highlights promising methods that support automatic parameterization and configuration coming from the fields of *machine learning* (ML) and *natural language processing* (NLP). Special focus is made on approaches that integrate with the commonly used evolutionary optimization methods: *genetic algorithms* (GAs; Holland, 1992) and *genetic programming* (GP; Koza, 1992). Hence, information items that are needed for parameterizing and configuring GA and GP systems in different design phases are identified.

In this paper, reference is made to the *VDI 2221* design method proposed by the Association of German Engineers (Verein Deutscher Ingenieure, 1993). It describes design processes in the context of product development that considerably builds on early editions of Pahl et al. (2007). The general applicability of these processes spans a wide range of industries, for example, electrical design, mechanical design, and software design. In the following section procedure as to how the method's elements map to the definition of design exploration and evolutionary optimization is shown.

The paper is organized as follows: the basis is built for a common understanding of design exploration in light of the design phases of VDI 2221. On the basis of this, applications of evolutionary optimization within each phase of design exploration are reviewed. This is followed by a discussion of the effort for configuring and parameterizing these approaches. Given this, systems that integrate methods from the fields of ML and NLP for configuration and parameterization are further reviewed. The paper closes with a conclusion on the surveyed works.

2. PHASES OF DESIGN EXPLORATION

At this stage a series of definitions is provided that frame the above statement from Pahl et al. (2007) more precisely. The following statements are mainly based on VDI 2221 and Pahl et al. (2007).

The *method* defines multiple design phases (cf. Fig. 1 and Table 1), each of which has two aspects of problem solving. First, every design phase represents a task to achieve a *process objective*, that is, to transform one product description into another. The plans for doing so are given by the design method. Second, each phase addresses one or more problem-solving steps of the overall design problem/task, that is, the search for requirement-satisfying product designs.

Product development starts with the phase referred to as *requirement modeling* or *task clarification*. The process objective of this phase is to formulate a *requirements specification* according to the client's desired results of product develop-

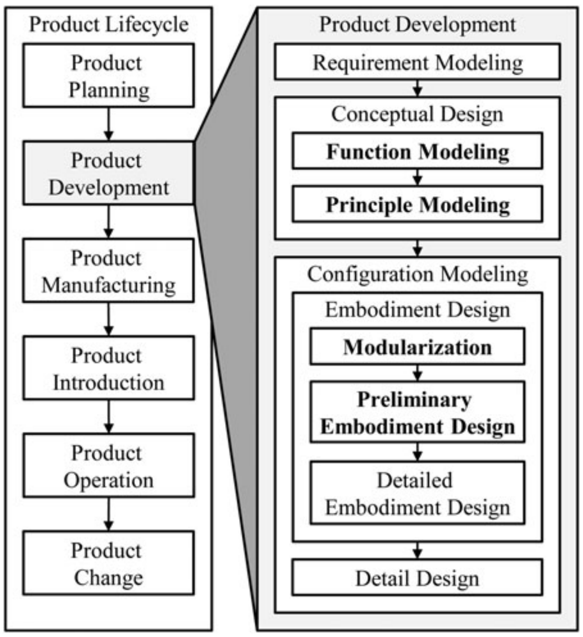

Fig. 1. Phases of product development in the context of the product life cycle, based on Pahl et al. (2007) and VDI 2221.

ment, where the latter is typically documented in the form of a *design brief*. Requirement modeling is also often revisited during later phases to change or add requirements that follow from intermediate design results.

Conceptual design consists of two phases, *function modeling* and *principle modeling*. Function modeling's process objective is to establish a more detailed requirement description by decomposing functional requirements into so-called function structures. These are models that provide a detailed description of the product's purpose by explaining how its subfunctions interact with each other. In VDI 2221 and Pahl et al. (2007), these interactions are conceptualized as conversion processes of energy, material, and signal flows within the product. The next phase, principle modeling, searches for adequate *solution principles* that realize a function structure's subfunctions. Solution principles describe the physical effect for achieving a function together with properties of material and geometry. For instance, the solution principle "winch" can be used to achieve the function "lift weight." The underlying physical effect is the mechanical advantage, which is applied in terms of the material and geometric properties of a rope, spool, and crank assembly. Finally, it has to be described how the solution principles are combined to work together as a whole. This kind of model is called a *principle solution* (also termed *working structure* or *concept*).

Defining properties of geometry, material, programming, and manufacturing is subject to *configuration modeling*. It comprises *embodiment design* and *detail design*. Embodiment design itself can be divided into three phases: *modularization, preliminary embodiment design*, and *detailed embodiment design*. First in line, modularization assigns the elements of the principle solution to structures of realizable

Table 1. *Role of design phases in solving the overall design task*

	Process Objective		Problem Solving Steps of Overall Design Task					
Phase	Given State	Desired State	Problem Analysis	Problem Formul.	System Synth.	System Analysis	Evaluation	Decision
Requirement modeling*	Design brief	Requirements specification	•	•				
		Conceptual Design						
Function modeling*	Req. spec.	Function structures	•	•				
Principle modeling*	Req. spec. and function structure	Principle solutions			•	•	•	•
		Configuration Modeling						
Embodiment design								
Modularization*	Req. spec. and principle solution	Modular structures	•	•				
Preliminary*	Req. spec. and modular structure	Preliminary layouts			•	•	•	•
Detailed	Req. spec. and preliminary layout	Definitive layout			•	•	•	•
Detail design	Req. spec. and definitive layout	Product documentation					•	•

*The phases of design exploration.

modules, that is, the act of modeling systems and subsystems. Then preliminary embodiment design concentrates on defining the most important modules with respect to geometry, material, and/or programming in terms of *preliminary layouts*. In detailed embodiment design, the separately designed modules of a preliminary layout are integrated to form a *definitive layout*. The last phase then is to prepare and document all details needed for manufacturing.

The early design phases from function modeling to preliminary embodiment design require the generation of a *set of alternatives* as their desired state. These phases and the recurring phase of requirement modeling are considered to be phases of design exploration.

3. EVOLUTIONARY DESIGN EXPLORATION

This section provides an overview of the usage of GAs and GP for design exploration in light of the discussed design phases. On the basis of this review, the parameterization and configuration of such applications will be described later.

3.1. Requirement modeling

The main task of requirement modeling is to establish a definition of the overall design task. Therefore, the demands and wishes that are stated in a design brief first have to be translated from a customer-oriented perspective to concrete technical requirements. The resulting list of requirements is further refined in order to resolve contradictions, elaborate details, determine the priority of requirements, and match the interests of the engineering company.

To the authors' knowledge uses of GA/GP-based methods in the field of requirement modeling are sparse. Nonetheless, there is a growing interest in search-based requirements optimization (Zhang et al., 2008).

3.1.1. Quality function deployment (QFD)

QFD is a method that assists in the translation between customer-oriented and technology-oriented perspectives on the design problems. QFD considers the requirements to be formulated in the language of the customer. These customer requirements are then correlated with technical attributes of the solution (Mehrjerdi, 2010). The common model used for QFD is the *house of quality* (HoQ; see Fig. 2). Within this context, GAs have been used to maximize satisfaction with customer requirements along with further goals (cost, design time, market share). This is done by searching for optimal targets for technical attributes, that is, determine the degree of attainment for each technical attribute (Tang et al., 2002), find target values for technical attributes (Bai & Kwong, 2003), determine importance weights for each technical attribute (Liu, 2010), or choose target values under consideration of competing products (Kwong et al., 2011). In all cases simple decision variable vectors are used for representing individuals. Evaluation relies on preference weights for customer requirements and two correlation matrices. One traces requirements to technical attributes (center of HoQ). The other defines dependencies among technical attributes (roof of HoQ).

3.1.2. Next release problem (NRP)

The NRP originates from software engineering. Software systems are typically deployed as a series of subsequent revi-

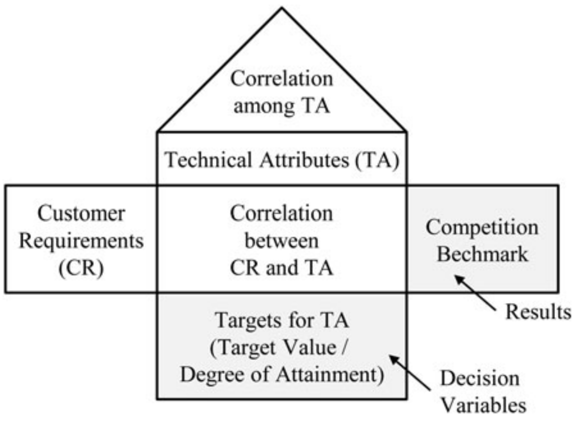
Correlation among TA
Technical Attributes (TA)
Customer Requirements (CR)
Correlation between CR and TA
Competition Bechmark
Results
Targets for TA (Target Value / Degree of Attainment)
Decision Variables

Fig. 2. House of quality model used for quality function deployment, based on Liu (2010).

sions. Given a set of customer requirements and a limited number of resources that a software company can invest in implementing requirements for the next revision, the NRP asks for the ideal allocation of resources to requirements (Bagnall et al., 2001). Greer and Ruhe (2004) apply a GA to find an optimal schedule that determines which of the requirements will be realized and in what release they will be realized. Later, Zhang et al. (2007) compared different GA variants on a multiple-objective formulation of the NRP, that is, minimizing cost while maximizing customer satisfaction (also called value). Like in QFD, decision variable vectors are used for representing individuals. Evaluation is also fairly simple because it mainly builds on attributes associated with requirements and customers.

3.1.3. *Optimize specific requirement models*

Over the span of different engineering disciplines, there also exist different and very specialized models for representing requirements. In the context of naval vessel design Sutcliffe et al. (2002) make use of so-called scenario task sequences (STS) for defining required interactions among human agents and computer equipment in a defined environment. A GA is employed to find optimal STS with respect to their influence on system reliability. In comparison to QFD and NRP, where decision variables could be readily used for defining genotypes, STS require a more complex chromosome representation. Sutcliffe et al. (2002) developed a mapping scheme, which encodes the sequence of tasks and the task–agent–equipment type combinations as binary string. For evaluation, a custom "performance interpreter" component models the expected influence of training, task complexity, operational time, quality, and utility on system reliability. Therefore, the interpreter component relies on attributes associated with agent and equipment types and rules that map these attributes to reliabilities.

Table 2 summarizes the evaluation criteria of the mentioned works, that is, the used objective and constraint functions.

3.2. Function modeling

The goal of function modeling is to clearly state the purpose of the design object, called *primary function*. This is done by delineating what subfunctions are needed to establish the primary function and how these functions must interoperate with each other, a process known as *functional decomposition*. Usually graph-based notations are used for describing function structures. In VDI 2221 and Pahl et al. (2007), for instance, functions are represented as nodes of a graph. These nodes are connected by directed edges that represent energy, material, and signal flows. The resulting graph depicts how forms of energy, material, and signals are transformed along the paths in order to enable the contained functions.

However, few works addressed optimization in context of function modeling. This may be because these models are very abstract in nature and lack commonly agreed schemes for their evaluation.

3.2.1. *Optimization of function structure*

Nonetheless, Güroğlu (2005), Jin and Li (2006), and Sun and Yao (2012) applied GP for functional decomposition. The standard tree-based representation of GP individuals offers a way to capture topological information in function structures. Hence, different tree serializations of function structures have been developed (as an example, see Fig. 3). In order to evaluate the generated function structures a number of abstract criteria have been proposed that require further explanation (see Table 3).

3.2.2. *Model consistency*

The GP-generated function structures should be consistent in light of the underlying model of functional decomposition, for example, with respect to the one of Pahl et al. (2007). Hence, a function structure must not be an arbitrary graph; it should rather correspond to the functional requirements for which it has been generated. For instance, the input and output flows of a function structure should match those of a more abstract black-box representation. Güroğlu (2005) and Jin and Li (2006) term this feasibility. Sun and Yao (2012) do not explicitly define such a measure. They require that their algorithm should include all *resources* that are mentioned in the requirements specification. However, these resources actually resemble the flows within the function structure.

3.2.3. *Uniqueness of elements and function structure size*

Sun and Yao (2012) define creativity as a matter of *variety*. Thus, GP trees with a relatively large amount of unique nodes are preferred. Jin and Li (2006) adopt a similar variety measure as part of a compound *desirability* score. Desirability also includes *function structure dimension*. The latter aims at limiting the size of elements in a function structure. Güroğlu (2005) also combines uniqueness of elements and function structure size. Their *complexity* measure is based on

Table 2. *Evaluation criteria used for requirement modeling*

Reference	Customer Satisfaction	Cost	Design Time Urgency	Market Share	Realization Dependencies	Reliability
Tang et al. (2002)	•	•			•	
Liu (2010)	•	•	•		•	
Kwong et al. (2011)	•			•	•	
Bai & Kwong (2003)	•				•	
Greer & Ruhe (2004)	•	•	•		•	
Zhang et al. (2007)	•	•				
Sutcliffe et al. (2002)						•

information entropy and promotes individuals with many unique elements.

3.2.4. *Change from initial population*

Both Güroğlu (2005) and Jin and Li (2006) compare generated function structures with individuals of the initial population. However, their assumptions on this comparison differ. Güroğlu (2005) generates an initial population from existing function structures made by engineers. The goal is to produce new unseen solutions from this set. Therefore, a *creativity* score measures the probability of functions and flows occurring with respect to the initial population. Jin and Li (2006), in turn, generate their initial population by means of a *graph grammar*. Hence, the generated graphs are per se artificial and do not have to correspond to existing products. The graph grammars rules follow the principles of functional decomposition and produce reasonable structures. Hence, the aim of Jin and Li (2006) is to keep graph grammar-produced structures where possible.

3.2.5. *Physical compatibility*

With reference to the following design phase of principle modeling, Jin and Li (2006) and Sun and Yao (2012) evaluate function structures by inspecting physical properties of their later realizations. As can be seen below, Jin and Li (2006) also search means combinations that can be used for realizing a function structure. The degree of compatibility among these means serves as a performance indicator of function structures. Sun and Yao (2012) adopt principles from the theory of inventive problem solving (Altshuller, 1999). Following the theory of inventive problem solving, a solution is inventive if physical contradictions have to be overcome. Hence, Sun and Yao (2012) introduce a score to determine inventive solutions based on the amount of physical contradictions being introduced by the function structure.

3.3. Principle modeling

Principle modeling is concerned with the search for ways to physically realize the defined functions. A rather abstract description of such means are the already mentioned solution principles (Pahl et al., 2007). As a result of the later design phases, solution principles will be implemented in concrete product components (e.g., in the context of ship design the solution principle "combustion engine" may be realized as a "four-stroke diesel engine"). Hence, especially in routine design tasks, it is also common to directly consider already realized product components at this stage of concept development.

Compared to requirement modeling and function modeling, more research has been done on using optimization for principle modeling. In this category three usage patterns for optimization can be identified.

3.3.1. *Optimization of principle solution combination*

Jin and Li (2006) partly base their approach on the method of Pahl et al. (2007) and search for solution principles that can be used for realizing a function structure and/or fit some given requirements. Jin and Li (2006) used a GA to search for a working structure that fits a given function structure. The particular function structure is evolved in parallel making use of GP as mentioned earlier. While the topology is defined by the function structure, the combination of solution principles is built from a library of solution principles. It provides informa-

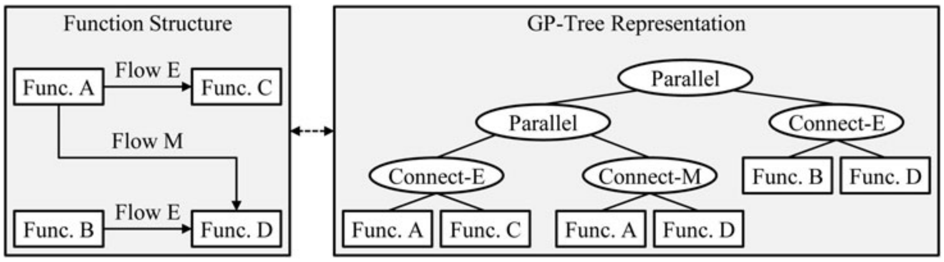

Fig. 3. Exemplary function structure to genetic programming tree serialization (Func. = Function), based on Jin and Li (2006).

Table 3. *Evaluation criteria used for function modeling*

Reference and Concept	Model Consistency	Uniqueness of Elements	Function Structure Size	Change From Initial Population	Physical Compatibility
Güroðlu (2005)					
Feasibility	•				
Complexity		•	•		
Creativity				•	
Jin & Li (2006)					
Function/hierarchical/ global flow consistency	•				
Function structure dimension			•		
Function variety		•			
Grammar rule usage				•	
Function-means compatibility					•
Sun & Yao (2012)					
Variety		•			
Contradictions					•

tion on available principles, including possible functions that can be implemented with a principle and constraints for integrating principles with others. The evaluation of solution principles follows compatibility, performance, and preference measures. An early step in the same direction is the work of Pham and Yang (1993). Here, so-called building blocks are combined to produce principle solutions for transmission systems. These building blocks constitute abstract solution principles like "helical gear pairs" or "belt-wheels." A GA performs the search over possible arrangements of building blocks considering their physical compatibility. Further, Campbell et al. (1999) proposed a seminal agent-based method named *A-design* that uses a GA-like method to search for optimal conceptual designs. A-design takes a user-defined functional specification as input and builds a configuration of abstract component descriptions from it. For instance, "gear," "shaft," or "spring" are typical abstract components. Their descriptions hold information about their connectivity with other components, that is, how energy and signals are transformed and how energy variables relate to others. In a second step, these abstract versions are instantiated to concrete components from a catalog, for example, "Spring: ERS-A1-36 \$0.93, K=16.0lb/in."

3.3.2. *Optimization of component combination*

Other approaches skip this level of abstraction and directly search for adequate components, whose descriptions are usually stored in a product database. This avoids the need to model abstract solution principles and offers direct ways for evaluation, for example, by using the price, weight, and dimensions of components. Xu et al. (2006) used genetic search to find appropriate component combinations for so-called function groups. These are groups of similarly used functions from existing example products and constitute a product platform's functionality. Function groups are correlated with both requirements (via *key characteristics*) and concrete components of the example products. This trace offers a flexible way of ranking and constraining component configurations with respect to requirements (cost, weight, manufacturing time, etc.). Hutcheson et al. (2006) started from a manually designed function structure and defined relevant component families for each function. Genetic search has been used to search for optimal component combinations with respect to cost, performance, and stress factors.

3.3.3. *Optimization of parametric product templates*

A third pattern is to use domain-specific key parameters (e.g., number of engines for a plane). This approach is prevalent in domains where there is no need to consider the variability of custom-made function structures. To a great extent the kind of functions to be fulfilled, their topology, and the means used for realizing them are all considered to be well known. The remaining variability is bound to an enumerable set of predefined product templates. Special discrete variables are then used to choose from this set (e.g., wing type), while additional continuous variables (e.g., wing span) refine the concept. Several works from the field of aerospace design followed this approach to identify configurations for aircraft designs and space operations with respect to mission requirements (Roth & Crossley, 1998; Crossley, 1999; Blasi et al., 2000; Perez & Behdinan, 2002; Hassan & Crossley, 2003). Further examples of this kind range from golf club design (Qiu et al., 2002) through hydropower systems (Parmee, 1998) to dynamics of bridge design (Wang & Tang, 2011). A significantly different parameter-oriented approach is proposed by Wu et al. (2008). They start from a function structure and propose a method that yields a *bond graph* model representing the physical parameters of solution principles. Genetic search is used to determine the optimal parameters for this model.

Table 4. *Evaluation criteria used for principle modeling*

Reference	Physical Compatibility	Cost	Weight	Grade Level	Designer Preference	Quality, Efficiency, Performance	Manufac. Time	Reliability Stress	Suffice Key Character.
Jin & Li (2006)	•	•	•	•	•				
Pham & Yang (1993)	•								
Campbell et al. (1999)	•	•	•			•			
Xu et al. (2006)		•	•				•	•	•
Hutcheson et al. (2006)		•				•		•	
Blasi et al. (2000)		•							•
Perez & Behdinan (2002)		•							•
Crossley (1999)			•			•			
Roth & Crossley (1998)			•						•
Hassan & Crossley (2003)	•		•					•	
Parmee (1998)	•	•							
Wang & Tang (2011)		•						•	•
Wu et al. (2008)	•								•

Table 4 summarizes the criteria used for evaluation. These criteria heavily rely on attributes that are associated with solution principles or components. In simple cases, these attributes are directly matched against required key characteristics, most often in terms of constraint functions. Prevalent properties of this kind are cost, weight/mass, reliability/stress, and other indicators of quality, efficiency, or performance. In other cases, properties of the overall solution are computed from its solution principles or components, for example, by means of equations that take various properties of multiple parts of the solution into account. Examples for this are precision, respond speed, or domain-specific criteria like orbital altitude or the number of satellites (Crossley, 1999). A central point in principle modeling is that a chosen working structure is actually physically feasible. Wu et al. (2008), for instance, ensure this through behavioral equations of bond graphs. Further optimization goals mentioned are designer preference, grade level of technology, and manufacturing time.

3.4. Modularization

In modularization working structures are split into interconnected subsystems that constitute the *system architecture*. Each of these so-called modules is defined by its specific set of attributes (e.g., nose/cabin/tail length of an aircraft's fuselage module). Preferably, modules are formed with respect to reoccurring patterns within the working structures of one or more products. Grouping these commonalities yields reusable modules that in turn enable product developers to increase variety, shorten lead times, and reduce costs (Simpson, 2004). Hence, modularization has received a lot of attention from the domain of product family design. Building on Fujita (2002) and taking recent developments into account, three types of optimization problems can be distinguished in this field.

1. *Optimization of module attributes:* The task is to optimize attribute parameterizations for a given system architecture. This problem has already been discussed for single products in the previous section. Optimization can also be utilized to find design parameters that best suit all members of a set of products. Considering requirements that hold for the whole product family (e.g., manufacturing concerns), this may lead to different results than optimizing each product individually. In this sense, the goal of Ferguson et al. (2009) was to optimize multiple car configurations simultaneously by means of GAs. One of their goals was to determine the optimal degree of adaptability for each configuration while considering the cost and performance of the whole product platform.
2. *Optimization of module combination:* This problem is about finding an optimal combination of predefined modules whose attribute parameterizations are fixed. This has also been outlined in the previous section when choosing adequate existing components for a single product. In light of a product family, it is desirable to share components among products. Again, taking

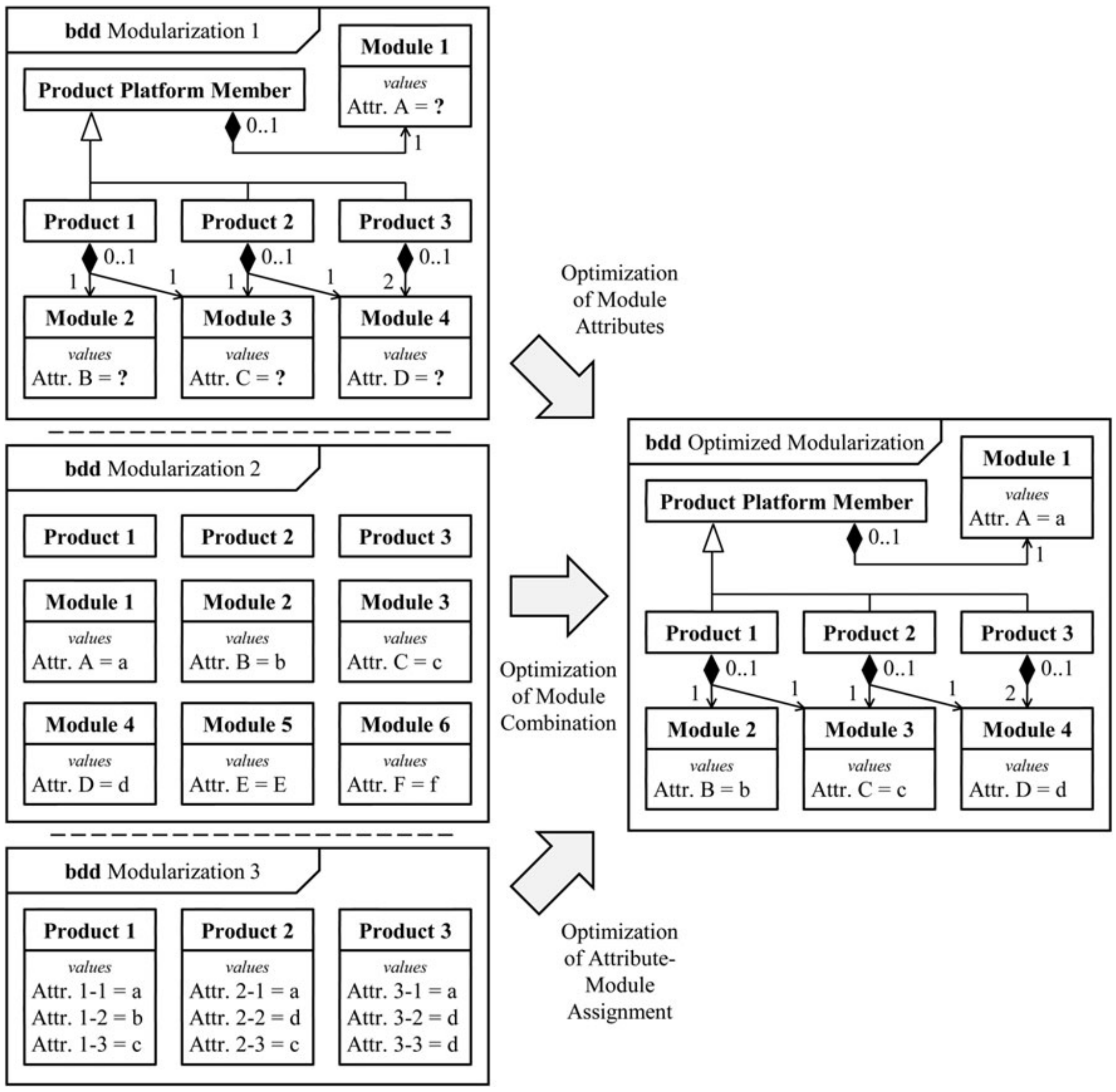

Fig. 4. Exemplification of typical modularization optimization problems.

product family-wide requirements into account may have effects on the optimization results. Within this context, Li and Azarm (2002) generate alternatives for a cordless screwdriver, then select common components from these for a product platform. Wang et al. (2011) provide a numerical example to demonstrate the feasibility of their GA-based approach for finding optimal module combinations.

3. *Optimization of attribute-module assignment:* Here, the goal is to find an optimal modularization scheme from given attribute parameterizations. After the parameters of each product have been determined, commonalities among the parameterizations can be identified and utilized to group identically parameterized product parts into modules. For this purpose, Simpson and D'Souza (2004) and Lewis et al. (2011) used GAs to assign attributes to modules. They had a specific interest in identifying those attributes that should be part of a common product platform.

Figure 4 illustrates all three optimization problems with respect to a desired product architecture. The right-hand side shows the desired modularized product architecture in *SysML* notation (Object Management Group, 2012). The left-hand side depicts the different starting points of the mentioned optimization approaches. Besides this, Table 5 provides an overview of the evaluation criteria used in the above works.

3.5. Preliminary embodiment design

The following phase of preliminary embodiment design addresses the geometrical realization of the system architecture. The goal is to find geometrical structures that embed the previously defined modules. Kicinger et al. (2005*a*) provide a

Table 5. *Evaluation criteria used for modularization*

Reference	Complexity	Market Share	Consumer Choice Utility	Profit	Cost	Product Level Problem	Performance	Adaptability	Commonality
Ferguson et al. (2009)					•	•	•	•	
Li & Azarm (2002)		•	•	•					
Wang et al. (2011)	•	•	•						
Lewis et al. (2011)						•			•
Simpson & D'Souza (2004)						•			•

comprehensive survey on evolutionary structural design and categorize prevalent problems in this field as follows:

3.5.1. Topology optimization

Topology optimization is also known as *layout optimization* or *topological optimum design*. Although the principle topology has already been set through functional decomposition and modularization, it has to be clarified how the parts have to be connected geometrically, taking account of weight, structural stability, and further aspects. Chapman et al. (1994), Bentley and Kumar (1999), Jakiela et al. (2000), Wang and Tai (2005), and Chen and Chiou (2013) consider a fixed geometrical grid defining the design space and employ a GA to determine which cells of the grid should be occupied by the product. Hamda et al. (2002) in turn used *Voronoi diagrams* consisting of filled and empty cells to indicate what space should be occupied. Besides these Schoenauer (1996) also takes so-called H-representations into account. These denote void by overlapping elementary shapes (e.g., rectangles) while all remaining parts of space are occupied by the product. Schoenauer (1996) compares all three forms of representation using GAs. Tai and Akhtar (2005) propose a graph-based representation that is consecutively rendered as a geometrical fill/no-fill grid. This graph holds information about a *b-spline*-based line model and thickness values for these lines. Another distinct group of works builds topologies from truss and frame elements. Here, GAs (Koumousis & Georgiou, 1994; Bohnenberger et al., 1995; Rajan, 1995; Nakanishi & Nakagiri, 1996; Rajeev & Krishnamoorthy, 1997; Azid et al., 2002; Mahdavi & Hanna, 2003; Kicinger et al., 2005*b*) and GP (Soh & Yang, 2000; Yang & Soh, 2002) are utilized to search for optimal connections of such elements.

3.5.2. Shape optimization

After the topology has been set, the next step is to determine the product shape. With respect to the nodes of a layout, this means that connections between nodes remain fixed while their positions are adjusted. For this purpose, evolutionary optimization has been applied to find adequate shapes based on *splines* (Kita & Tanie, 1999; Cerrolaza et al., 2000; Annicchiarico & Cerrolaza, 2001; Muc & Gurba, 2001), *polygons* (Grierson & Pak, 1993; Keane & Brown, 1996; Soh & Yang, 1996), *boundary representations* (Woon et al., 2001; Namgoong et al., 2012), and *constructive solid geometry* (Bentley & Wakefield, 1996).

3.5.3. Sizing optimization

The last problem under consideration for design exploration is identifying optimal sizes of product parts. Specific dimensional aspects with strategic influence have already been in focus of modularization. The current stage takes account of more detailed matters of sizing. Just to mention a few exemplary works, sizing optimization with evolutionary algorithms has been addressed in context of determining sizes of basic elements like columns, beams, trusses, and rods (Hajela, 1990; Deb, 1991; Rajeev & Krishnamoorthy, 1997; Muc & Gurba, 2001; Kaveh & Kalatjari, 2002; Jármai et al., 2003) or domain-specific sizing problems like cross-sectional dimensions in bridge design (Jenkins, 1992), or bay and floor sizing in building design (Rafiq & Rustell, 2011).

3.5.4. Unified embodiment optimization

Some works combine the mentioned embodiment aspects in a unified process. Schnier and Gero (1996) "grow" floor plan layouts using a shape grammar. Here, all possible modifications are predefined by production rules. A GA then searches for application sequences of rules that result in solutions showing high similarity with a set of given design cases. Kicinger et al. (2005*c*) proposed another rule-based system that uses GAs to jointly search for the optimal topology and shape of a building's structural system, and Hornby (2004) use a similar method for table design. Further, Turrin et al. (2011) and Gerber and Lin (2012) integrate parametric architectural design with a GA. Hence, topology, shape, and size may be jointly optimized with respect to the underlying parametric model. In addition to these works, a very different approach is taken by Shang et al. (2009). They render conceptual design as a form of geometrical decomposition toward so-called functional surfaces. Initially, functional requirements are mapped to primitive geometric objects that form a prototype concept. After being codified as genotype, this concept is evolved to complex geometric structures using a genetic search. The engineer then performs an evaluation of generated solutions manually.

Despite the variety of approaches to preliminary embodiment design, the criteria used for evaluation show significant

Table 6. *Entities and relations from requirements (A) or design rationale (B) that are used for problem specification*

Phase	Entities	Relations
Requirement modeling		
QFD	A: CR	A: preference weights of CR
	A: Competing products	A: target values of competing products
	B: TA and their target values	B: engineering characteristics (e.g., costs, design times) of TA target values
		B: correlations between CR and TA
		B: correlations between TA
NRP	A: customers	A: importance weights of customers
	A: requirements	B: costs of requirements
		A: values and urgencies of requirements for customers
		B: dependencies between requirements
STS	A: agents	A: weights of properties
	A: tasks	B: attributes of agents and equipment
	A: equipment	B: evaluation rules
Function modeling	A: initial function model (black box)	A: weights of evaluation criteria
	B: functions	B: engineering characteristics of flows
	B: flows	B: operators for functional decomposition (e.g., graph grammar rules)
		B: validity rules for combining functions
		B: taxonomies for functions/flows
Principle modeling	A: evolved function model (function structure)	B: map between functions and SP/components
	B: SP	B: map between SP and components
	B: components	B: engineering characteristics (weight, cost, grade level, designer preference, etc.) of SP/components
	B: product template model	B: validity rules for combining SP/components
		B: models for input/output behaviors
Modularization	A: modules (incl. product platforms)	B: engineering characteristics of products/modules
	A: products	A: target values for characteristics
		B: map between modules and products
Preliminary embod. design	A: existing topology	A: geometric limitations (e.g. boundaries, grids, degrees of freedom)
	A: materials	B: models for physical behaviors
	B: geometric primitive objects	B: operators for geometric modification (e.g., shape grammar rules)
		B: engineering characteristics of materials/objects

Note: QFD, Quality function deployment; CR, customer requirements; TA, technical attributes; NRP, next release problem; STS, scenario task sequences; SP, solution principles.

commonalities. Most often the goals are to achieve a design of minimum volume or mass while maintaining structural stability (in terms of stress, displacement, buckling, deflection, vibration, etc.). Another prevalent requirement is to satisfy certain geometrical affordances like symmetry (Kicinger et al., 2005*b*) or boundary conditions (Annicchiarico & Cerrolaza, 2001). Further, more domain-specific evaluation criteria comprise usable space (Rafiq & Rustell, 2011), material constraints (Mahdavi & Hanna, 2003), light characteristics (Bentley & Wakefield, 1996), aerodynamic properties (Namgoong et al., 2012), and costs for manufacturing and operation (Deb, 1991; Jenkins, 1992; Rafiq & Rustell, 2011; Gerber & Lin, 2012; Namgoong et al., 2012).

4. CONFIGURATION AND PARAMETERIZATION

The reviewed works show how the design problems of a specific design phase can be formulated in such a way so that evolutionary optimization methods become applicable. In most cases, exemplary scenarios demonstrated how the methods could be applied. However, in order to be applied to any other practical problem instance, one has to prepare design variables $\boldsymbol{x}$, constants $\boldsymbol{c}$, and functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$, $g_i(\boldsymbol{x}, \boldsymbol{c})$, $h_j(\boldsymbol{x}, \boldsymbol{c})$ for the specific scenario at hand. In simple cases, only constants have to be adjusted (parameterization), while more complex adjustments require the preparation of variables and functions (configuration). Taking a closer look at the items used for parameterizing and configuring the above problems (cf. Table 6), four recurring patterns can be observed.

In Pattern 1, the representation of individuals used for GA/GP is predominantly determined by the desired description of the design phase's process objective (e.g., to yield a function structure). At this point, the design variables have to be defined and subsequently transformed into the chromosome or GP tree representation of the particular approach. Typically, this requires the user to define the variable ranges under consideration (e.g., in case of function modeling, the variables may represent the available functions to choose from).

In Pattern 2, representation stands in close relation to the methods used for fitness evaluation. These methods require individuals to have certain attributes from which evaluation criteria may be calculated (e.g., cost or weight of a component). Hence, the design space has to be attributed beforehand, either by translating requirements to attributes, like in QFD, or by drawing upon experience from existing design rationale; that is, it is well known to the engineer what attributes typically need to be considered for the problem at hand. Conversely, the calculation of an individual's fitness is affected by the chosen definitions for design variables. Objective functions need to be configured to incorporate new design variables or modified variable ranges.

In Pattern 3, the fitness of individuals is further determined by the given descriptions of the preceding design phase (e.g., the chosen working principles should match the functions of a given function structure). Typically, method-specific objective functions need to be parameterized by setting constants (e.g., weights setting the relative importance of requirements). Nonetheless, often completely new objective functions need to be added to reflect custom requirements (e.g., a simulation model for estimating the range of an aircraft design, where the desired range has been determined by the preceding principle modeling phase). The same applies to constraint functions. In GA/GP-based optimization, these are often reflected as penalty objective functions. In this case, comparable actions need to be done for parameterization and configuration. Instead of using penalty functions, constraints can also be considered in advance during the generation of individuals. Here, the same parameterization and configuration steps apply.

In Pattern 4, some works also require specifying relations needed for the generation of individuals (e.g., rules that define feasible design decisions within functional decomposition). Though such relations are out of the scope of the optimization problem's definition, they are valuable for improving the problem-solving efficiency of evolutionary optimization methods. Here, the design space is restricted by integrating rules for production and/or validation into the mechanisms for generating individuals. This corresponds to the idea of using constraints to limit the individuals that are considered for evaluation to only feasible ones.

ML and NLP approaches offer a variety of ways to discover information entities and relations from available data. In case of ML, it is usually assumed that this data is already available in a formal format (e.g., as vector or graph representation), whereas NLP methods address the extraction of information entities and relations from naturally written texts. Principally, both fields provide methods to explore a space of possible hypotheses for models that best explain the available data. The next sections review literature where this characteristic has been employed to support parameterization and/or configuration of optimization problems. Moreover, reference is made to promising ML/NLP approaches that could contribute to this task. Therefore, symbols used for design variables and constants are both considered to be information entities. Among these entities, relations are defined. They represent range definitions, value assignments, objective functions, constraint functions, or relations used for generation. Table 6 lists the information entities and relations that were used for configuration and parameterization considering the reviewed works. They are further distinguished whether they originate from product specific requirements (A) or from existing knowledge for decision making in that phase, that is, the design rationale (B).

4.1. Methods supporting parameterization

Objectives and constraints are often formulated in a general sense and must be adapted to the concrete problem at hand before the optimization problem can be solved. For instance, imagine the task of finding an optimal bridge design. The original problem for that task could state that "the length of a bridge must be greater than the width of the river it spans." Now, in order to find designs that suffice this constraint, the width of the river must be defined. Such concretizations of the problem are typically modeled by constants $\boldsymbol{c}$ that configure the objective functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = [f_1(\boldsymbol{x}, \boldsymbol{c}), f_2(\boldsymbol{x}, \boldsymbol{c}), \ldots, f_k(\boldsymbol{x}, \boldsymbol{c})]^T$ and/or constraint functions $g_i(\boldsymbol{x}, \boldsymbol{c})$, $h_j(\boldsymbol{x}, \boldsymbol{c})$.

Cluster-oriented genetic algorithms provide a method for dynamically parameterizing objective functions (Parmee & Bonham, 1999; Parmee, 2002). They are a variant of GA that allows for exploring design spaces that have not been considered in the first place. This is supposed to support early design stages where the problem definition is uncertain and subject to frequent redefinition. A threshold constant on each objective function is introduced that determines which candidates will be considered in the solution set. The threshold can be adapted iteratively, where low values broaden the search while high values cause a more focused exploration of distinct regions of high performance. These regions are visually depicted so that the user can change the threshold in order to drive the problem definition toward a desired direction. Parmee and Bonham (1999) demonstrate this by showing the high-performance regions of three objective functions in a multiple-objective optimization problem. Two of the regions do overlap, which indicates that there is a subset of compromise solutions for the associated objectives. However, the high-performance region of the third objective is isolated from this. Relaxing the threshold for this objective is equivalent to a reduction of its relative importance and causes the associated region to expand. This ultimately leads to an overlapping of all three regions.

Ren and Papalambros (2011) tackle the problem of designer preference elicitation for design optimization by using a customized version of the *efficient global optimization* algorithm. It integrates a *support vector machine* (SVM) to iteratively refine constraints used for optimization. With each iteration of the efficient global optimization algorithm, the human designer chooses relatively suitable candidates from the generated solutions. From this response, the SVM classification mechanism produces a decision function that is used by a GA to sample new candidates from the design space. Completing the cycle, these samples are then presented to the designer to gather new feedback. Hence, the SVM decision function resembles a configurable constraint that is adapted over time in order to produce optimal solutions as desired by the designer.

4.2. Methods supporting configuration

Parameterization can be considered as choosing an instance from a rather narrow set of optimization problems that is defined by a known problem class. Configuration instead addresses the definition of that problem class. In the next sections different approaches are reviewed that help to determine all further elements of the problem definition beside constants, that is, design variables, objective functions, and constraint functions.

4.3. Product attribute extraction

Ghani et al. (2006) characterized the term *product attribute extraction*. The idea of this NLP method is to produce formal representation of a product's attributes from semistructured or unstructured product descriptions (e.g., product recessions). Ideally, the extraction produces a variable description of each attribute found in the document corpus with its range defined by the written values. Each product description can then be represented by a data vector made up of the found attributes. This information can be directly used to define design variables $\boldsymbol{x}$ for product attributes. In contrast, product attribute extraction establishes the link between products/components and their attributes. In cases where design variables represent products/components, this relationship can be used to formulate objective functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$ and/or constraint functions $g_i(\boldsymbol{x}, \boldsymbol{c})$, $h_j(\boldsymbol{x}, \boldsymbol{c})$ that evaluate their characteristics.

Besides the original supervised method for extracting product attributes of Ghani et al. (2006), semisupervised (Wu et al., 2009) and unsupervised (Wong et al., 2008) methods have been developed. These systems operate on semistructured product descriptions like websites, where additional context information is given by the structure of the HTML document. Raju et al. (2009) focused on unsupervised methods that can also be used for product attribute extraction from unstructured texts.

4.4. Clustering solutions with similar attributes

In product attribute extraction, the link between a variable (design object) and its range (characteristics of design object) is established by translating the information that is already explicit in documents into a formal representation. A different group of approaches targets the case where this information is not explicitly given. The starting point for these approaches is a set of solutions to a design problem. These solutions are then grouped with respect to the similarity of their characteristics using clustering methods. Each identified group is considered as the class of its contained instances. Having identified such class–instance relationships, the information can be used to express each class as a design variable $\boldsymbol{x}$ where the instances represent the variable's range.

Besides this, the inspection of discovered solution groups can provide hints for refining the optimization problem. Constraint functions $g_i(\boldsymbol{x}, \boldsymbol{c})$, $h_j(\boldsymbol{x}, \boldsymbol{c})$ may be defined that reflect the value boundaries of a specific group in order to formulate an optimization problem that specifically targets that region. Overlapping solution groups, in turn, suggest that more decisive optimization goals should be added to the set of objective functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$.

Xu et al. (2006) used a neural network to group components of similar products, called fan filter units, with respect to their functionality. Their work presents a way that design variables and their ranges can be defined automatically. Therefore, each component is described by means of a data vector that holds information about its material, signal, and energy flows. Using this as training data, the neural network is trained to group components that may be named differently but fulfill the same functionality; for example, components named "fan filter unit control," "monitoring," and "working condition control" are assigned to the group "control." The resulting groups stand for abstract concepts that denote the functions of a product platform, for example, "control," "insulation," and "air filtering." For the next step, each group is considered a design variable with its range defined by the associated components. This enables the GA-based optimization to identify new optimal component combinations for the product platform's set of functions.

Veerappa and Letier (2011) use a distance-based hierarchical clustering algorithm in the domain of requirement modeling. Their underlying problem is finding a set of requirements that maximizes value while minimizing cost. For each solution pair on the value- and cost-dimensioned Pareto frontier, a distance score measures their difference in fulfilled requirements. From these distances, a hierarchy of solution clusters is formed. On the one hand, it is supposed that navigating the hierarchy eases the selection of desired solutions. Moving from a cluster to one of its subclusters narrows the set of considered solutions. At this stage, in order to explore this region of interest in greater detail, one may introduce new constraints that correspond to the subcluster's boundary and rerun the search process. On the other hand, the proposed method also supports the identification of different solutions that achieve similar objectives. This is the case when clusters overlap. As a consequence for problem formulation, further objective functions may be introduced.

Similarly Reich and Fenves (1991) used hierarchical clustering to group characteristics of bridge designs (material, lanes, span, etc.) with respect to their occurrence in example design variants. The resulting hierarchical structure resembles a taxonomy of bridge designs, where each subclass is characterized by some specific properties (e.g., wooden bridges and steel constructions form two distinct groups).

4.5. Discovering relationships

The configuration of optimization problems can also be supported by making the relations among design variables, constants, objective functions, and constraint functions more tangible. It is essential for the formulation of optimization problems to know how these elements can be combined and what engineering concepts they reflect. Different approaches exist to acquire this information automatically:

- Analyze how the elements interact with each other during optimization.
- Examine what elements were used in combination over different problem formulations.
- Extract semantic relations among elements from design documentation.

Innovization is a method that discovers regressed functions over the design variables, objectives, and constraints (Bandaru & Deb, 2013). These so-called design principles express hidden regularities within the problem formulation like proportionality between design variables. In order to find design principles, the discovery itself is formulated as an optimization problem searching over possible combinations of design variables, objectives, and constraints. The problem is then solved using a GA. As a preparatory step, innovization uses clustering to group related solutions. Bandaru and Deb (2013) distinguish between *lower level* and *higher level innovization*. Lower level innovization is used to identify design principles that are common to a subset of solutions from a single trade-off front. The extracted design principles can be used to identify solution subsets that show a desired property or conversely to find properties that characterize a desired region. Higher level innovization targets uncertainty in problem definition. Therefore, different trade-off fronts from varying problem parameterizations are fed into the regression to determine how changes to the problem affect prevalent design principles. For instance, it allows investigating what parts of the problem remain unaffected from the addition or deletion of new constraints.

The goal of Matthews et al. (2006) was to discover abstract relations among the elements used for formulating optimization problems. They used a GP-based approach to determine what elements in a problem specification are strongly related to each other. This strength is computed from the covariances that are computed over the values of existing solutions. From the discovered relations, heuristics like "the thickness of a design has a strong influence on its weight" are derived.

Sarkar et al. (2008) analyzed multiple formulations of an optimization problem using the feature extraction method *singular value decomposition*. By means of singular value decomposition, latent syntactic relations among the used variables, constants, and functions were discovered. The trained meta-model can be queried to determine what variables, constants, or functions are relevant in the presence of one another. This feature can be used to guide engineers in finding an adequate ensemble of constraint and objective functions.

Another group of NLP works can be utilized to guide the creation of constraint or objective functions by showing how relevant concepts are related to each other. Wong et al. (2011) extended the product attribute extraction approach of Wong et al. (2008) with a clustering method that is applied on the found product attributes. It arranges semantically related attributes in hierarchically ordered groups. For instance, the attributes "charger" and "battery" of a digital camera are associated with the group "power information," which is likewise a member of the group "hardware information." Dong and Agogino (1997) analyzed documents from the field of product development to determine what central terms are mentioned in common contexts like "valve" and "pressure." Similarly Yang et al. (2005) automatically derived a thesaurus of design terms. Liang et al. (2012) extracted terms that are characteristic for product components such as "print cartridge" and "ink drop generator" for an inkjet printhead design. They also present a method to identify sentences within product development documentation that are used for describing problems or solutions. Li and Ramani (2007) proposed an extraction method that uses some predefined *ontology* that represents domain knowledge about the texts to be analyzed. Given this, it is possible to build more sophisticated application ontologies that contain detailed information on the relations of mentioned concepts.

4.6. Computing objective functions

ML methods were also used to determine the components of objective functions $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = [f_1(\boldsymbol{x}, \boldsymbol{c}), f_2(\boldsymbol{x}, \boldsymbol{c}), \ldots, f_k(\boldsymbol{x}, \boldsymbol{c})]^T$. Instead of describing each objective function manually, regression or classification methods may be used to derive objective functions from existing examples of good design solutions.

Both Kwong et al. (2011) and Yang and Chen (2014) use regressed functions to describe the relation between customer requirements and engineering characteristics in the context of QFD. This information is crucial for assessing the potential customer satisfaction with the chosen engineering characteristics. The known satisfaction with existing products has been used to obtain estimates for this objective function.

In addition to these approaches, there has been an ongoing interest in the field of engineering design to discover relations between the structural and behavioral properties of designs. Such methods could be used to automate the formulation of fitness functions. Ivezic and Garrett (1998) employed artificial neural networks (ANNs) to determine a regressed function. This function predicts behavioral aspects (e.g., deflection) and realization characteristics (e.g., construction time) of bridge designs from their structural properties (e.g., geometry, material, expected load). Using knowledge of the expected behavior, designers are able to evaluate whether the design objectives can be met. Reich and Barai (1999) and Neocleous and Schizas (2002) also used ANNs. Their goal was to determine relations between the characteristics of propeller designs and their performance. Skibniewski et al. (1997) and Szczepanik et al. (1995) proposed rule induction methods that predict the manufacturing effort of beam designs and estimate the weight of designs.

4.7. Learning metamodels for evaluation

The goal is to identify an alternative objective function $\boldsymbol{f}'(\boldsymbol{x}, \boldsymbol{c})$ that is equivalent or at least approximate to $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$. This new

Table 7. *Summary of machine learning/natural language processing approaches to the support of parameterization and configuration*

Concept (References)	Patterns	Target of Config./Param.			
		Constants	Design Variables	Objective Func.	Constraint Func.
Iteratively adjust constants of					
Objective functions (Parmee & Bonham, 1999; Parmee, 2002)	3	•			
Constraint functions (Ren & Papalambros, 2011)	3	•			
Extract information on design variables and associated attributes from design documentation (Ghani et al., 2006; Wong et al., 2008; Raju et al., 2009; Wu et al., 2009)	1, 2		•	•	•
Group attributed solutions to discover					
Design variables (Xu et al., 2006)	1		•		
New objective and constraint functions (Reich & Fenves, 1991; Veerappa & Letier, 2011)	2			•	•
Identify which problem elements belong together by					
Analyzing how problem elements interact with each other during optimization (Matthews et al., 2006; Bandaru & Deb, 2013)	1, 2, 3		•	•	•
Examining what elements were used in combination over different problem formulations (Sarkar et al., 2008)	1, 2, 3		•	•	•
Extracting semantic relations among elements from design documentation (Dong & Agogino, 1997; Yang et al., 2005; Li & Ramani, 2007; Wong et al., 2011; Liang et al., 2012)	1, 2, 3		•	•	•
Computing objective functions by					
Relating engineering characteristics to requirements (Kwong et al., 2011; Yang & Chen, 2014)	3			•	
Relating structural attributes to behavioral attributes (Szczepanik et al., 1995; Skibniewski et al., 1997; Ivezic & Garrett, 1998; Reich & Barai, 1999; Neocleous & Schizas, 2002)	3			•	
Learning metamodels for evaluation in order to					
Replace costly evaluation functions (Perez & Behdinan, 2002; Lin, 2003)	2			•	
Broaden the applicability of evaluation functions by abstracting from existing relations (Bhatta & Goel, 1994; Chabot & Brown, 1994; Reffat & Gero, 2000)	2			•	
Restrict the feasible design space with respect to identified					
Chunks (Moss et al., 2004; Mukerjee & Dabbeeru, 2012)	4	•	•		•
Design rules (Gero et al., 1994; Forouraghi, 1999; Hanna, 2007; Yogev et al., 2010)	4	•	•		•

objective function models the details inherent in the components of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = [f_1(\boldsymbol{x}, \boldsymbol{c}), f_2(\boldsymbol{x}, \boldsymbol{c}), \ldots, f_k(\boldsymbol{x}, \boldsymbol{c})]^T$ on a more abstract level. Here, the underlying aim is either to apply the same functions to a wider range of problems or to perform the optimization more efficiently. In certain domains, for instance, using exact evaluation functions for the selection of individuals can become very costly, for example, if simulations are to be computed or marketing interviews have to be conducted. Therefore, more abstract evaluation schemes like performance heuristics or approximations can help to make solving the problem more practicable.

Perez and Behdinan (2002) and Lin (2003) both employed ANNs to learn a classification from existing evaluations. If enough evaluation results have been generated, then the neural network is trained and used as a replacement for the costly evaluation function.

There is a range of further ML applications that learn abstract meta-models from existing descriptions of relations.

A *knowledge compilation* approach was taken by Chabot and Brown (1994). Their system learned the abstract restriction "x surrounds y" from relations among the diameters of different components. Bhatta and Goel (1994) used *instance-based learning* to generalize functional relations of two designs to yield an abstract principle solution. Given a functional and behavioral description of a sulfuric acid cooler and a nitric acid cooler, the model of a generic heat exchanger is derived. Using a hierarchical clustering method (Reffat & Gero, 2000) grouped geometric restrictions (e.g., symmetry, adjacency) with respect to their joint use within floor plan designs. In addition, Wang and Shan (2006) provide a comprehensive overview over metamodeling techniques in optimization that go beyond the herein discussed ML/NLP approaches.

4.8. Learning patterns of good design

Here, the idea is to investigate the inverse relation of the overall objective function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$. ML methods are used to determine combinations of value assignments for subsets of $\boldsymbol{x}$ that cause $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$ to drop considerably. These promising value combinations are called *chunks*. Introducing a bias toward the selection of known chunks can help to solve similar optimization problems better and/or faster. This bias can be implemented in different ways:

- Change some variables of $\boldsymbol{x}$ to constants $\boldsymbol{c}$ whose values correspond to those of a desired chunk.
- Introduce additional constraints $g_i(\boldsymbol{x}, \boldsymbol{c})$ and/or $h_j(\boldsymbol{x}, \boldsymbol{c})$ that restrict $\boldsymbol{x}$ to value combinations of known chunks.
- Use special generation methods that tend to choose values of known chunks for $\boldsymbol{x}$.

Moss et al. (2004) extended the A-Design method (Campbell et al., 1999) to identify well performing patterns in solutions. These chunks are then reused in new problems to generate candidates during optimization. As a result, the system's efficiency and effectiveness are incrementally increased. Mukerjee and Dabbeeru (2012) use dimension reduction to identify chunks. In this case, the learned chunks decide whether a design is feasible or not. By analyzing communication protocols of designers, their approach was able to assign a natural language symbol to these learned relations (e.g., the parts of a construction "fit tight").

Finding appropriate designs for a set of required engineering characteristics has been in the scope of various ML-based approaches. Starting from the problem of finding a frame geometry with optimal load characteristics, Hanna (2007) applied the SVM technique to learn a classification that conversely identifies adequate frame constructions for given loads. Forouraghi (1999) used multivariate regression trees to determine variant groups that optimally fulfill a goal function. These groups define ranges for design variables that should be obeyed in order to optimize the objective functions. Both Gero et al. (1994) and Yogev et al. (2010) draw upon the idea that ML can be seen as a search for hypotheses that best explain the data. They make use of GAs to search for production rules for a shape grammar. The language produced by the shape grammar then defines the set of reasonable shapes to choose from.

Table 7 provides a summary of the discussed approaches to configuration and parameterization. It further depicts how these relate to the patterns of problem formulation that could be observed from the reviewed applications of evolutionary optimization.

5. CONCLUSION

We presented a short yet comprehensive overview of GA/GP-based engineering design optimization approaches from various phases of design exploration, that is, requirement modeling, function modeling, principle modeling, modularization, and preliminary embodiment design. Although several commonalities can be observed, each of the discussed problems was required to be set up according to the requirements and designing rationale that are specific for the project and engineering domain. In order to integrate this spectrum in a computational framework that spans multiple design phases, additional mechanisms are needed for facilitating the automatic configuration and parameterization of optimization problems.

Despite the different nature of the design phases, the formulation of related optimization problems still reduces to defining constants, design variables, objective functions, and constraint functions. Recurring patterns in problem formulation could be observed from the reviewed applications of evolutionary optimizations. This in turn suggested that methods targeting these patterns can support the formulation of optimization problems over multiple design phases.

Various methods were reviewed from the fields of ML and NLP for this purpose. They can be utilized to acquire the information entities and relations that are needed for configuration and parameterization. Up to now, several of these methods have been integrated into the representation, evaluation, and generation aspects of GA/GP-based design optimization. As discussed, there are further applications of ML/NLP in the field of engineering design that could be used to expand this support.

REFERENCES

Altshuller, G. (1999). *The Innovation Algorithm: TRIZ, Systematic Innovation and Technical Creativity*. Worcester: Technical Innovation Center.

Annicchiarico, W., & Cerrolaza, M. (2001). Structural shape optimization 3D finite-element models based on genetic algorithms and geometric modeling. *Finite Elements in Analysis and Design 37(5)*, 403–415.

Azid, I.A., Kwan, A.S.K., & Seetharamu, K.N. (2002). An evolutionary approach for layout optimization of a three-dimensional truss. *Structural and Multidisciplinary Optimization 24(4)*, 333–337.

Bagnall, A.J., Rayward-Smith, V.J., & Whittley, I.M. (2001). The next release problem. *Information and Software Technology 43(14)*, 883–890.

Bai, H., & Kwong, C.K. (2003). Inexact genetic algorithm approach to target values setting of engineering requirements in QFD. *International Journal of Production Research 41(16)*, 3861–3881.

Bandaru, S., & Deb, K. (2013). Higher and lower-level knowledge discovery from Pareto-optimal sets. *Journal of Global Optimization 57(2)*, 281–298.

Bentley, P., & Kumar, S. (1999). Three ways to grow designs: a comparison of evolved embryogenies for a design problem. *Proc. 1st Genetic and Evolutionary Computing Conf., GECCO'99*. San Francisco, CA: Morgan Kaufmann.

Bentley, P.J., & Wakefield, J.P. (1996). Conceptual evolutionary design by a genetic algorithm. *Engineering Design and Automation 2(3)*, 119–131.

Bhatta, S.R., & Goel, A.K. (1994). Discovery of physical principles from design experiences. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8(2)*, 113–123.

Blasi, L., Iuspa, L., & Del Core, G. (2000). Conceptual aircraft design based on a multiconstraint genetic optimizer. *Journal of Aircraft 37(2)*, 351–354.

Bohnenberger, O., Hesser, J., & Männer, R. (1995). Automatic design of truss structures using evolutionary algorithms. *Proc. 1995 IEEE Int. Conf. Evolutionary Computation, ICEC'95*. New York: IEEE.

Campbell, M.I., Cagan, J., & Kotovsky, K. (1999). A-Design: an agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design 11(3)*, 172–192.

Cerrolaza, M., Annicchiarico, W., & Martinez, M. (2000). Optimization of 2D boundary element models using b-splines and genetic algorithms. *Engineering Analysis With Boundary Elements 24(5)*, 427–440.

Chabot, R., & Brown, D.C. (1994). Knowledge compilation using constraint inheritance. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8(2)*, 125–142.

Chapman, C.D., Saitou, K., & Jakiela, M.J. (1994). Genetic algorithms as an approach to configuration and topology design. *Journal of Mechanical Design 116(4)*, 1005–1012.

Chen, T.Y., & Chiou, Y.H. (2013). Structural topology optimization using genetic algorithms. *Proc. 2013 World Congr. Engineering, WCE'13*. Hong Kong: Newswood Limited.

Crossley, W.A. (1999). Optimization for aerospace conceptual design through the use of genetic algorithms. *Proc. 1st NASA/DoD Workshop on Evolvable Hardware*. Washington, DC: IEEE Computer Society.

Deb, K. (1991). Optimal design of a welded beam via genetic algorithms. *AIAA Journal 29(11)*, 2013–2015.

Dong, A., & Agogino, A.M. (1997). Text analysis for constructing design representations. In *Artificial Intelligence in Design '96* (Gero, J.S., & Sudweeks, F., Eds.), pp. 21–38. Dordrecht: Kluwer Academic.

Ferguson, S., Kasprzak, E., & Lewis, K. (2009). Designing a family of reconfigurable vehicles using multilevel multidisciplinary design optimization. *Structural and Multidisciplinary Optimization 39(2)*, 171–186.

Forouraghi, B. (1999). On utility of inductive learning in multiobjective robust design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 13(1)*, 27–36.

Fujita, K. (2002). Product variety optimization under modular architecture. *Computer-Aided Design 34(12)*, 953–965.

Gerber, D., & Lin, S.-H.E. (2012). Designing-in performance through parameterization, automation, and evolutionary algorithms: "H.D.S. BEAGLE 1.0." *Proc. 17th Int. Conf. Computer-Aided Architectural Design Research in Asia, CAADRIA'12*, pp. 141–150, Chennai, India, April 25–28, 2012.

Gero, J.S., Louis, S.J., & Kundu, S. (1994). Evolutionary learning of novel grammars for design improvement. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8(2)*, 83–94.

Ghani, R., Probst, K., Liu, Y., Krema, M., & Fano, A. (2006). Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter 8(1)*, 41–48.

Greer, D., & Ruhe, G. (2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology 46(4)*, 243–253.

Grierson, D.E., & Pak, W.H. (1993). Optimal sizing, geometrical and topological design using a genetic algorithm. *Structural and Multidisciplinary Optimization 6(3)*, 151–159.

Güroğlu, S. (2005). *An Evolutionary Methodology for Conceptual Design*. Ankara: Middle East Technical University Press.

Hajela, P. (1990). Genetic search—an approach to the nonconvex optimization problem. *AIAA Journal 28(7)*, 1205–1210.

Hamda, H., Jouve, F., Lutton, E., Schoenauer, M., & Sebag, M. (2002). Compact unstructured representations for evolutionary design. *Applied Intelligence 16(2)*, 139–155.

Hanna, S. (2007). Inductive machine learning of optimal modular structures: estimating solutions using support vector machines. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 21(4)*, 351–366.

Hassan, R.A., & Crossley, W.A. (2003). Multi-objective optimization of communication satellites with two-branch tournament genetic algorithm. *Journal of Spacecraft and Rockets 40(2)*, 266–272.

Holland, J.H. (1992). Genetic algorithms. *Scientific American 267(1)*, 66–72.

Hornby, G.S. (2004). Functional scalability through generative representations: the evolution of table designs. *Environment and Planning B: Planning and Design 31(4)*, 569–587.

Hutcheson, R.S., Jordan, Jr., R.L., Stone, R.B., Terpenny, J.P., & Chang, X. (2006). Application of a genetic algorithm to concept variant selection. *Proc. ASME 2006 Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf., IDETC/CIE'06*. New York: ASME.

Ivezic, N., & Garrett, Jr., J.H. (1998). Machine learning for simulation-based support of early collaborative design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(2)*, 123–139.

Jakiela, M.J., Chapman, C., Duda, J., Adewuya, A., & Saitou, K. (2000). Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering 186(2–4)*, 339–356.

Jármai, K., Snyman, J.A., Farkas, J., & Gondos, G. (2003). Optimal design of a welded I-section frame using four conceptually different optimization algorithms. *Structural and Multidisciplinary Optimization 25(1)*, 54–61.

Jenkins, W.M. (1992). Plane frame optimum design environment based on genetic algorithm. *Journal of Structural Engineering 118(11)*, 3103–3112.

Jin, Y., & Li, W. (2006). Design concept generation: a hierarchical coevolutionary approach. *Journal of Mechanical Design 129(10)*, 1012–1022.

Kaveh, A., & Kalatjari, V. (2002). Genetic algorithm for discrete-sizing optimal design of trusses using the force method. *International Journal for Numerical Methods in Engineering 55(1)*, 55–72.

Keane, A.J., & Brown, S.M. (1996). The design of a satellite boom with enhanced vibration performance using genetic algorithm techniques. *Proc. 2nd Conf. Adaptive Computing in Engineering Design and Control, ACEDC'96*, pp. 107–113, Plymouth, UK, March 26–28.

Kicinger, R., Arciszewski, T., & De Jong, K. (2005*a*). Evolutionary computation and structural design: a survey of the state-of-the-art. *Computers and Structures 83(23–24)*, 1943–1978.

Kicinger, R., Arciszewski, T., & De Jong, K. (2005*b*). Evolutionary design of steel structures in tall buildings. *Journal of Computing in Civil Engineering 19(3)*, 223–238.

Kicinger, R., Arciszewski, T., & De Jong, K. (2005*c*). Parameterized versus generative representations in structural design: an empirical comparison. *Proc. 7th Genetic and Evolutionary Computing Conf., GECCO'05*. New York: ACM.

Kita, E., & Tanie, H. (1999). Topology and shape optimization of continuum structures using GA and BEM. *Structural and Multidisciplinary Optimization 17(2–3)*, 130–139.

Koumousis, V.K., & Georgiou, P.G. (1994). Genetic algorithms in discrete optimization of steel truss roofs. *Journal of Computing in Civil Engineering 8(3)*, 309–325.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Kwong, C.K., Luo, X.G., & Tang, J.F. (2011). A methodology for optimal product positioning with engineering constraints consideration. *International Journal of Production Economics 132(1)*, 93–100.

Lewis, P.K., Murray, V.R., & Mattson, C.A. (2011). A design optimization strategy for creating devices that traverse the Pareto frontier over time. *Structural and Multidisciplinary Optimization 43(2)*, 191–204.

Li, H., & Azarm, S. (2002). An approach for product line design selection under uncertainty and competition. *Journal of Mechanical Design 124(3)*, 385–392.

Li, Z., & Ramani, K. (2007). Ontology-based design information extraction and retrieval. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 21(2)*, 137–154.

Liang, Y., Liu, Y., Kwong, C.K., & Lee, W.B. (2012). Learning the "whys": discovering design rationale using text mining—an algorithm perspective. *Computer-Aided Design 44(10)*, 916–930.

Lin, J.-J. (2003). Constructing an intelligent conceptual design system using genetic algorithm. *Journal of Materials Processing Technology 140(1–3)*, 95–99.

Liu, C.-H. (2010). A group decision-making method with fuzzy set theory and genetic algorithms in quality function deployment. *Quality & Quantity 44(6)*, 1175–1189.

Mahdavi, S.H., & Hanna, S. (2003). An evolutionary approach to microstructure optimisation of stereolithographic models. *Proc. 2003 Congr. Evolutionary Computation, CEC'03*. New York: IEEE.

Marler, R.T., & Arora, J.S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization 26(6)*, 369–395.

Matthews, P.C., Standingford, D.W.F., Holden, C.M.E., & Wallace, K.M. (2006). Learning inexpensive parametric design models using an augmented genetic programming technique. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 20(1)*, 1–18.

Mehrjerdi, Y.Z. (2010). Quality function deployment and its extensions. *International Journal of Quality & Reliability Management 27(6)*, 616–640.

Moss, J., Cagan, J., & Kotovsky, K. (2004). Learning from design experience in an agent-based design system. *Research in Engineering Design 15(2)*, 77–92.

Muc, A., & Gurba, W. (2001). Genetic algorithms and finite element analysis in optimization of composite structures. *Composite Structures 54(2–3)*, 275–281.

Mukerjee, A., & Dabbeeru, M.M. (2012). Grounded discovery of symbols as concept–language pairs. *Computer-Aided Design 44(10)*, 901–915.

Nakanishi, Y., & Nakagiri, S. (1996). Optimization of frame topology using boundary cycle and genetic algorithm. *JSME International Journal Series A: Solid Mechanics and Material Engineering 39(2)*, 279–285.

Namgoong, H., Crossley, W.A., & Lyrintzis, A.S. (2012). Morphing airfoil design for minimum drag and actuation energy including aerodynamic work. *Journal of Aircraft 49(4)*, 981–990.

Neocleous, C.C., & Schizas, C.N. (2002). Artificial neural networks in estimating marine propeller cavitation. *Proc. 2002 Int. Joint Conf. Neural Networks, IJCNN'02*. New York: IEEE.

Object Management Group. (2012). *OMG System Modeling Language (OMG SysML™)*. Needham: Object Management Group.

Pahl, G., Beitz, W., Feldhusen, J., & Grote, K.-H. (2007). *Pahl/Beitz Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung*, 7th ed. Berlin: Springer.

Parmee, I.C. (1998). Evolutionary and adaptive strategies for efficient search across whole system engineering design hierarchies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(5)*, 431–445.

Parmee, I.C. (2002). Improving problem definition through interactive evolutionary computation. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 16(3)*, 185–202.

Parmee, I.C., & Bonham, C.R. (1999). Cluster-oriented genetic algorithms to support interactive designer/evolutionary computing systems. *Proc. 1999 Congr. Evolutionary Computation, CEC'99*. New York: IEEE.

Perez, R.E., & Behdinan, K. (2002). Effective multi-mission aircraft conceptual design optimization using a hybrid multi-objective evolutionary method. *Proc. 9th AIAA/ISSMO Symp. Multidisciplinary Analysis and Optimization*. Reston, VA: AIAA.

Pham, D.T., & Yang, Y. (1993). A genetic algorithm based preliminary design system. *Journal of Automobile Engineering 207(2)*, 127–133.

Qiu, S.L., Fok, S.C., Chen, C.H., & Xu, S. (2002). Conceptual design using evolution strategy. *International Journal of Advanced Manufacturing Technology 20(9)*, 683–691.

Rafiq, M.Y., & Rustell, M.J.F. (2011). Building information modelling driven by the evolutionary computing. *Proc. 18th European Group for Intelligent Computing in Engineering Workshop, EG-ICE'11*, Twente, The Netherlands, July 6–8.

Rajan, S.D. (1995). Sizing, shape, and topology design optimization of trusses using genetic algorithm. *Journal of Structural Engineering 121(10)*, 1480–1487.

Rajeev, S., & Krishnamoorthy, C.S. (1997). Genetic algorithms-based methodologies for design optimization of trusses. *Journal of Structural Engineering 123(3)*, 350–358.

Raju, S., Shishtla, P., & Varma, V. (2009). A graph clustering approach to product attribute extraction. *Proc. 4th Indian Int. Conf. Artificial Intelligence, IICAI'09*, pp. 1438–1447, Tumkur, India, December 16–18.

Reffat, R.M., & Gero, J.S. (2000). Computational situated learning in design. In *Artificial Intelligence in Design '00* (Gero, J.S., Ed.), pp. 589–610. Dordrecht: Kluwer Academic.

Reich, Y., & Barai, S.V. (1999). Evaluating machine learning models for engineering problems. *Artificial Intelligence in Engineering 13(3)*, 257–272.

Reich, Y., & Fenves, S.J. (1991). The formation and use of abstract concepts in design. In *Concept Formation Knowledge and Experience in Unsupervised Learning* (Fisher, Jr., D.H., Pazzani, M.J., & Langley, P., Eds.), pp. 323–353. San Francisco, CA: Morgan Kaufmann.

Ren, Y., & Papalambros, P.Y. (2011). A design preference elicitation query as an optimization process. *Journal of Mechanical Design 133(11)*, 111004-1 –111004-9.

Roth, G.L., & Crossley, W.A. (1998). Commercial transport aircraft conceptual design using a genetic algorithm based approach. *Proc. 7th AIAA/USAF/NASA/ISSMO Symp. Multidisciplinary Analysis and Optimization*. Reston, VA: AIAA.

Sarkar, S., Dong, A., & Gero, J.S. (2008). Learning symbolic formulations in design optimization. *Proc. 3rd Int. Conf. Design Computing and Cognition, DCC'08*. Dordrecht: Springer Science+Business Media.

Schnier, T., & Gero, J.S. (1996). Learning genetic representations as alternative to hand-coded shape grammars. In *Artificial Intelligence in Design '96* (Gero, J.S., & Sudweeks, F., Eds.), pp. 39–57. Dordrecht: Kluwer Academic.

Schoenauer, M. (1996). Shape representations and evolution schemes. *Proc. 5th Annual Conf. Evolutionary Programming, Evolutionary Programming*. Cambridge, MA: MIT Press.

Schon, D.A. (1992). Designing as reflective conversation with the materials of a design situation. *Research in Engineering Design 3(3)*, 131–147.

Shang, Y., Huang, K.Z., & Zhang, Q.P. (2009). Genetic model for conceptual design of mechanical products based on functional surface. *International Journal of Advanced Manufacturing Technology 42(3–4)*, 211–221.

Simpson, T.W. (2004). Product platform design and customization: status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 18(1)*, 3–20.

Simpson, T.W., & 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)*, 119–129.

Skibniewski, M., Arciszewski, T., & Lueprasert, K. (1997). Constructability analysis: machine learning approach. *Journal of Computing in Civil Engineering 11(1)*, 8–16.

Soh, C.K., & Yang, J. (1996). Fuzzy controlled genetic algorithm search for shape optimization. *Journal of Computing in Civil Engineering 10(2)*, 143–150.

Soh, C.K., & Yang, Y. (2000). Genetic programming-based approach for structural optimization. *Journal of Computing in Civil Engineering 14(1)*, 31–37.

Sun, G., & Yao, S. (2012). A framework for an evolutionary computation approach to supporting concept generation. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 56(1)*, 1972–1976.

Sutcliffe, A., Chang, W.-C., & Neville, R. (2002). Optimizing system requirements with evolutionary computation. *Proc. 2002 Congr. Evolutionary Computation, CEC'02*. Washington, DC: IEEE Computer Society.

Szczepanik, W., Arciszewski, T., & Wnek, J. (1995). Empirical performance comparison of two symbolic learning systems based on selective and constructive induction. *Proc. Workshops at the 14th Int. Joint Conf. Artificial Intelligence, IJCAI'95*, pp. 203–214, Montreal, Canada, August 19–25, 1995.

Tai, K., & Akhtar, S. (2005). Structural topology optimization using a genetic algorithm with a morphological geometric representation scheme. *Structural and Multidisciplinary Optimization 30(2)*, 113–127.

Tang, J., Fung, R.Y.K., Xu, B., & Wang, D. (2002). A new approach to quality function deployment planning with financial consideration. *Computers & Operations Research 29(11)*, 1447–1463.

Turrin, M., von Buelow, P., & Stouffs, R. (2011). Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. *Advanced Engineering Informatics 25(4)*, 656–675.

Veerappa, V., & Letier, E. (2011). Understanding clusters of optimal solutions in multi-objective decision problems. *Proc. 19th IEEE Int. Requirements Engineering Conf., RE'11*. Washington, DC: IEEE Computer Society.

Verein Deutscher Ingenieure. (1993). *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte (VDI 2221)*. Berlin: Beuth.

Wang, G.G., & Shan, S. (2006). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design 129(4)*, 370–380.

Wang, H., Zhu, X., Wang, H., Hu, S.J., Lin, Z., & Chen, G. (2011). Multi-objective optimization of product variety and manufacturing complexity in mixed-model assembly systems. *Journal of Manufacturing Systems 30(1)*, 16–27.

Wang, L.-F., & Tang, A.-P. (2011). Collaborative optimization design of reinforcement concrete bridge considering aseismic requirements. *Proc. 2011 Int. Conf. Electric Technology and Civil Engineering, ICETCE'11*. New York: IEEE.

Wang, S.Y., & Tai, K. (2005). Structural topology design optimization using genetic algorithms with a bit-array representation. *Computer Methods in Applied Mechanics and Engineering 194(36–38)*, 3749–3770.

Wong, T.-L., Bing, L., & Lam, W. (2011). Normalizing web product attributes and discovering domain ontology with minimal effort. *Proc. 4th ACM Int. Conf. Web Search and Data Mining, WSDM'11*. New York: ACM.

Wong, T.-L., Lam, W., & Wong, T.-S. (2008). An unsupervised framework for extracting and normalizing product attributes from multiple web sites. *Proc. 31st Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval, SIGIR'08*. New York: ACM.

Woon, S.Y., Querin, O.M., & Steven, G.P. (2001). Structural application of a shape optimization method based on a genetic algorithm. *Structural and Multidisciplinary Optimization 22(1)*, 57–64.

Wu, B., Cheng, X., Wang, Y., Guo, Y., & Song, L. (2009). Simultaneous product attribute name and value extraction from web pages. *Proc. 2009 IEEE/WIC/ACM Int. Joint Conf. Web Intelligence and Intelligent Agent Technology, WI-IAT'09*. Washington, DC: IEEE Computer Society.

Wu, Z., Campbell, M.I., & Fernández, B.R. (2008). Bond graph based automated modeling for computer-aided design of dynamic systems. *Journal of Mechanical Design 130(4)*, 041102–041102-11.

Xu, Q.L., Ong, S.K., & Nee, A.Y.C. (2006). Function-based design synthesis approach to design reuse. *Research in Engineering Design 17(1)*, 27–44.

Yang, M.C., Wood III, W.H., & Cutkosky, M.R. (2005). Design information retrieval: a thesauri-based approach for reuse of informal design information. *Engineering With Computers 21(2)*, 177–192.

Yang, Z., & Chen, Y. (2014). Fuzzy optimization modeling approach for QFD-based new product design. *Journal of Industrial Engineering 2014*, 1–8.

Yang, Y., & Soh, C.K. (2002). Automated optimum design of structures using genetic programming. *Computers and Structures 80(18–19)*, 1537–1546.

Yogev, O., Shapiro, A.A., Member, S., & Antonsson, E.K. (2010). Computational evolutionary embryogeny. *IEEE Transactions on Evolutionary Computation 14(2)*, 301–325.

Zhang, Y., Finkelstein, A., & Harman, M. (2008). Search based requirements optimisation: existing work and challenges. In *Requirements Engineering: Foundation for Software Quality, 14th Int. Working Conf., REFSQ'08* (Paech, B., & Rolland, C., Eds.), LNCS Vol. 5025, pp. 88–94. Berlin: Springer.

Zhang, Y., Harman, M., & Mansouri, S.A. (2007). The multi-objective next release problem. *Proc. 9th Genetic and Evolutionary Computing Conf., GECCO'07*. New York: ACM.

Julian Eichhoff is a doctoral student at the Institute of Computer-Aided Product Development Systems at the University of Stuttgart. He received BS and MS degrees in computer science from Furtwangen University. His research focuses on integrating machine learning methods with computer-aided design, formal design representations, and design automation. From 2009 to 2012 he participated in the project Re-Use of Semantic Product Knowledge in New Product Design Processes under Prof. Wolfgang Maass at Furtwangen University.

Dieter Roller is a Professor, the Chair of Computer Science Fundamentals, and the Director of the Institute of Computer-Aided Product Development at the University of Stuttgart. He is an Honorary Professor at the University of Kaiserslautern and serves on the board of trustees of the Technische Akademie Esslingen. He is Chairman of several national symposia, congresses, and workshops in product development and automation. Dr. Roller gained comprehensive industrial experience as a former research and development manager with worldwide responsibility for computer-aided design technology within an international computer company.