Evolutionary Computation and Structural Design: a Survey of the State of the Art[*]

Rafal Kicinger[1]
Civil, Environmental and Infrastructure Engineering Department,
George Mason University, Fairfax, VA 22030, USA

Tomasz Arciszewski
Civil, Environmental and Infrastructure Engineering Department,
George Mason University, Fairfax, VA 22030, USA

Kenneth De Jong
Computer Science Department,
George Mason University, Fairfax, VA 22030, USA

Abstract
Evolutionary computation is emerging as a new engineering computational paradigm, which may significantly change the present structural design practice. For this reason, an extensive study of evolutionary computation in the context of structural design has been conducted in the Information Technology and Engineering School at George Mason University and its results are reported here. First, a general introduction to evolutionary computation is presented and recent developments in this field are briefly described. Next, the field of evolutionary design is introduced and its relevance to structural design is explained. Further, the issue of creativity/novelty is discussed and possible ways of achieving it during a structural design process are suggested. Current research progress in building engineering systems' representations, one of the key issues in evolutionary design, is subsequently discussed. Next, recent developments in constraint-handling methods in evolutionary optimization are reported. Further, the rapidly growing field of evolutionary multiobjective optimization is presented and briefly described. An emerging subfield of coevolutionary design is subsequently introduced and its current advancements reported. Next, a comprehensive review of the applications of evolutionary computation in structural design is provided and chronologically classified. Finally, a summary of the current research status and a discussion on the most promising paths of future research are also presented.

Keywords
Evolutionary computation, Structural design, Conceptual design, Multiobjective analysis, Optimization, Constraints, Computer aided design

1. Introduction

The new Millennium witnesses the emergence of Information Technology as the driving force behind the progress in civil engineering, particularly in the area of computation as related to design. The growing sophistication of computer programs, their availability, increased speed of computations and their ever-decreasing costs have already had a significant impact on civil engineering, and that can be considered a paradigm change.

Up to very recently, computers in structural design were used mostly for the analytical purposes in the detailed design stages. Nowadays, their role is becoming more and more versatile. They are being applied to all stages of the design process, from the generation of design concepts (design topologies, or layouts), through preliminary design (design shape specification), and finally in the detailed design process (sizing of structural members). That requires a new intellectual and computational framework to fully benefit from the progress in Information Technology. Among computational paradigms, evolutionary computation (EC) is now recognized as particularly appropriate for various traditional and novel computational applications in structural engineering.

The major objective of this paper is to present a comprehensive survey of the recent developments of evolutionary-based methods in structural engineering as well as to provide their historical context. Also,

[1] Corresponding author: E-mail: rkicinge@gmu.edu, Phone: (+1)703-993-1658, Fax: (+1)703-993-1521

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

a unified picture of evolutionary computation proposed by one of the authors [1] is discussed together with a discussion of the state-of-the-art (SOTA) in the EC research areas that are particularly relevant to structural design. Further, a summary of the current research status and a discussion on the most promising paths of future research is presented.

There are several other literature reviews on evolutionary techniques in structural engineering [2-7] but they are all too narrow, i.e. they either consider only a particular species of evolutionary algorithm (e.g. genetic algorithms), or focus on applications for a specific stage of a design process, e.g. conceptual design. None of these surveys attempts to provide as comprehensive and unified view of EC in structural engineering as it is intended in this paper. This survey provides a relatively complete picture of the research that has been done in the past and that is currently under way. It should become a useful reference for both newcomers as well as researchers already working in this field.

A significant effort was made to gather, analyze, and appropriately describe research developments in this field since its roots dating back to mid 1970's. It is, however, impossible to provide an exhaustive literature review discussing every piece of work that has been done over the years. Thus, the authors had to arbitrarily select the most representative work known to them (particularly, in the case of very prolific researchers). It is also possible that equally important and stimulating research unknown to the authors was unintentionally omitted.

The survey is structured in the following way. First, a general introduction to evolutionary computation is presented in section 2 and recent developments in this field are briefly described. Section 3 describes the field of evolutionary design and discusses its relevance to structural engineering. In this section, the issue of creativity/novelty is discussed and possible ways of achieving it during a structural design process are suggested. Section 4 contains an overview of recent developments in building engineering systems' representations, one of the key issues in evolutionary design particularly when creativity/novelty is sought. Since almost every structural design problem involves some kind of constraints, section 5 reports SOTA in constraint-handling methods used in evolutionary optimization. It is also common that a given structural design problem has multiple and often conflicting objectives. Section 6 discusses recent developments in the rapidly growing field of evolutionary multiobjective optimization. Section 7 provides an overview of the emerging subfield of coevolutionary design and discusses its potential for structural design. Section 8 contains a comprehensive literature review of the applications of evolutionary computation in structural design. A summary of the major applications since mid 1980's is provided in a chronological order and classified with respect to application domain, and major EC characteristics. Finally, a discussion on the current research status and most promising paths of future research is presented in section 9.

2. Evolutionary Computation

Evolutionary Computation (EC) is a modern search technique which uses computational models of processes of evolution and selection. Concepts and mechanisms of Darwinian [8] evolution and natural selection are encoded in evolutionary algorithms (EAs) and used to solve problems in many fields of engineering and science.

Strong resemblance to biological processes as well as their initial applications for modeling complex adaptive systems [9] influenced the terminology used by EC researchers. It borrows a lot from genetics, evolutionary theory and cellular biology. Thus, a candidate solution to a problem is called an *individual* while an entire set (or more accurately a superset) of current solutions is called a *population*. For some problem domains, a population may be broken into several *subpopulations*. The actual representation (encoding) of an individual is called its *genome* or *chromosome*. Each genome consists of a sequence of *genes*, i.e. attributes that describe an individual. A value of a gene is called an *allele*. When individual solutions are modified to produce new candidate solutions they are said to be *breeding* and the new candidate solution is called an *offspring* or a *child*. During the evaluation of a candidate solution, it receives a grade called *fitness*, which indicates the quality of the solution in the context of a given problem. When the current population is replaced by offspring, the new population is called a new *generation*. Finally, the entire process of searching for an optimal solution is called *evolution* [10].

2.1 Evolutionary Algorithms

Evolutionary algorithms are a family of population-based search algorithms that simulate the evolution of individual structures by interrelated processes of selection, reproduction, and variation. There is a variety of EAs that have been proposed and studied. They all share a common set of underlying assumptions but differ in the breeding strategy to be used and representation on which EAs operate.

Historically, three major EAs have been developed: Evolution Strategies (ES) [11,12], Evolutionary Programming (EP) [13], and Genetic Algorithms (GAs) [9]. These algorithms have been mostly used to evolve solutions to parameterized problem domains. On the other hand, the fourth major EA developed more recently, Genetic Programming (GP) [14], has been used to evolve actual computer programs to solve a number of computational tasks [10]. There are also many hybrid models incorporating various features of the above paradigms, including the CHC algorithm [15], the structured GA [16], the breeder GA [17], the messy GA [18], and many others.

From the engineering point of view, EC can be understood as a search and optimization process in which a population of solutions undergoes a process of gradual changes. This process depends on the fitness (a formal measure of perceived performance) of the individual solutions as defined by the environment (objective function).

A canonical EA consists of the following steps:

```
1. Initialize the population
2. Evaluate all members of the population
While the termination condition is not satisfied
{
    3. Select individual(s) in the population to be parent(s)
    4. Create new individuals by applying the variation operators to the copies of parent(s)
    5. Evaluate new individuals
    6. Replace some/all of the individuals in the current population with the new individuals
}
```

Before an actual evolutionary process begins, an initial population of individuals (solutions) is created. Traditionally, the initial population is created randomly but several other initialization techniques have also been used (e.g. starting from a set of previously known or arbitrarily assumed solutions). Next, each individual in the initial population is evaluated and assigned a fitness value.

Using the fitness scores, the selection mechanism chooses a subset of the current population as parents to create new individuals. When the selection mechanism uses bias toward individuals with better fitness, the created offspring will, more likely, have higher fitness. Once the set of parents has been selected, the new individuals are created by copying them and applying variation operators.

There are several commonly used selection strategies within EC community. *Fitness-proportional selection* [9] normalizes the fitness values of all individuals in the population and assigns these normalized values as probabilities that their respective individuals will be selected. *Ranked selection* works by first ranking all individuals in the population by their fitness, and use these ranks, rather than actual fitness values, to determine selection probabilities of the individuals. A common form of ranked selection is a *linear ranking* [19,20] where individuals are first sorted in an increasing order according to their fitness values. Each individual is then selected with a probability based on some linear function of its sorted rank. Another popular selection strategy is a *tournament selection*. In this strategy, a pool of *n* individuals is picked at random from the population. Each of the individuals in the pool is selected independently and it might be the case that the same individual will be selected multiple times. Next, an individual from the pool with highest fitness value is selected to form the new population. This procedure is repeated as many times as necessary to create either an entirely new population or a subset of it. The pool size is a parameter that controls the magnitude of the selection pressure. Finally, the *truncation selection* chooses only a certain proportion of the best individuals in the population. This strategy is most popular within the ES community, where it is used in two basic flavors: (μ, λ) and $(\mu+\lambda)$ [21]. In the former case, the selection operates on the offspring population only, whereas in the latter case it selects individuals from a joint population of both parents and offspring.

The two most popular variation operators are *mutation* and *recombination*. Mutation acts on a single individual and works by applying some variation to one or more genes in the individual's chromosome (similar to a variation operator used in other search mechanisms like hill climbing or simulated annealing). Recombination, on the other hand, operates on multiple individuals (usually two) and combines parts of these individuals to create new ones.

The newly created individuals are evaluated and assigned fitness values. Then, either all or only a subset of the current population is replaced by these new individuals. If the entire population is replaced by the new individuals then the algorithm is called *generational* EA. On the other hand, if only a subset of the original population is replaced then the algorithm is called a *steady-state* EA. Steps 3-6 of the canonical EA defined earlier are performed until an assumed stopping criterion is met, which is usually defined as an arbitrary number of generations or fitness function evaluations.

2.2 Evolutionary Computation and Engineering Design

This basic evolutionary process described above is called a ‘simple evolutionary algorithm’ in a sense that it contains the minimal set of features necessary to be a Darwinian evolutionary system. These simple EAs have surprisingly useful properties, primarily related to solving difficult global optimization problems. They perform well when applied to problems with nonlinear, stochastic, temporal, or chaotic components, where traditional optimization techniques, like gradient descent, hill climbing, and purely random search, are generally unsatisfactory. It is in this context that much of the work on engineering applications has taken place historically: using simple EAs for design optimization.
The three main issues in applying EAs to an engineering design problem are:

1. Selecting an appropriate representation for engineering designs.
2. Defining efficient genetic operators.
3. Providing an adequate evaluation function for estimating the “fitness” of generated solutions (points in the search space).

An appropriate representation of an engineering system is one of the most crucial elements of evolutionary design. This issue is particularly important when creativity/novelty of designs produced in evolutionary processes is one of the major goals. The process of creating an efficient and adequate representation of an engineering system for evolutionary design is complicated and involves elements of both science and art. One has to take into account not only important aspects of understanding traditional modeling of an engineering system, but also relevant computational issues that include search efficiency, scalability, and mapping between a search space (genotypic space) and a space of actual designs (phenotypic space). A more detailed discussion of EA representations is presented in section 4.

Appropriate choice and implementation of genetic operators, i.e. mutation and recombination opreators, and careful tuning of their rates is an important issue as it can have a big impact on the success of EAs and has therefore been a subject of both theoretical [22] as well as experimental investigations [23-25]. Any particular implementation of a mutation or recombination operator is representation dependent. Thus, for example GAs with binary string representations use the *bit-flip* mutation and 1-, or 2-point crossover, while ES with real-valued vectors use the *Gaussian* mutation and a recombination operator that swaps/averages parents’ alleles. Genetic operators are primary sources of exploration in EAs. On the other hand, selection mechanisms provide EAs with exploitative power. Thus, by properly defining and controlling the variation mechanisms (genetic operators), one can achieve a higher level goal of finding “an effective balance between further exploration of unexplored regions of the search space and exploiting the regions already explored.” [1].

Another important issue in successful application of EAs is to choose an adequate fitness evaluation function for a problem domain. Evaluation functions provide EAs with feedback about the fitness of each individual in the population. EAs use this feedback to bias the search process in order to improve the population’s average fitness. Naturally, the details of a particular fitness function are problem specific.

Table 1 provides a description of all commonly used EAs in terms of decisions that are made during an implementation of a particular EA. It is a modified table initially proposed in [6]. The particular decisions are summarized in terms of attributes and their values. Using this characterization, it is then straightforward to describe a given EA, e.g. a GA or ES, and its relationship to other EAs [6].

2.3 Advanced Evolutionary Algorithms

Various modern trends in EC relax some of the assumptions found in the canonical EAs. For example, in *multiobjective EAs*, a requirement of a single fitness value determining the quality of an individual is replaced by several independent fitness criteria. Another assumption of using a single evolving population is relaxed in *parallel*, or *distributed*, EAs as well as in *coevolutionary algorithms* (CEAs). In a fairly popular model of a parallel EA, called the *island-model* EA [26], evolution occurs in multiple parallel subpopulations evolving independently with occasional ‘migrations’ of some individuals among subpopulations. CEAs typically use multiple subpopulations but additionally modify another fundamental assumption, namely that individuals are no longer evaluated independently of one another. Two common models of CEAs include *cooperative CEAs* [27], where the fitness of an individual is assessed through ‘cooperation’ with individuals from other subpopulations, and *competitive CEAs* [28], where the fitness of an individual is determined by its competition against individuals from other populations. Coevolutionary EAs are discussed in more detail in section 7.

Next section presents a subfield of EC, called evolutionary design, which is directly related to engineering design problems. It also discusses the issues of creativity and emergence in engineering design processes.

Table 1: Attributes describing commonly used EA implementations.

No.	Attribute			Attribute Values				
				1	2	3	4	5
1	Solution representation	Encoding		Binary	Real-valued	Graph-based	Computer code	Other
		Length		Fixed	Variable			
2	Population initialization	Mechanism		Random generation	Selection from a group of known solutions	User defined		
		Population size		1	Fixed	Variable		
3	Parent selection mechanism			Truncation	Ranking	Fitness proportional	Tournament	Uniform
4	Variation mechanism	Mutation	Type	Bit-flip	Gaussian	Subtree	User defined	
			Rate	0	Fixed	Adaptive	Random	
		Crossover	Type	N-point	Swap	Uniform	Subtree	User defined
			Rate	0	Fixed	Adaptive	Random	
5	Survival selection mechanism			Truncation	Ranking	Fitness proportional	Tournament	Uniform

3. Evolutionary Design and Creativity

Evolutionary design is a branch of EC that integrates ideas from computer science (evolutionary algorithms), engineering (design science) and evolutionary biology (natural selection) to solve engineering design problems [29]. Four major categories of problems considered by evolutionary design include evolutionary design optimization, creative evolutionary design, evolutionary art, and evolutionary artificial life forms.

Common attributes shared by evolutionary techniques, which are relevant to engineering design processes include [30]:

- little, if any, a priori knowledge of the search environment
- excellent search capabilities due to efficient sampling of the design search space
- ability to avoid local optima
- ability to handle high dimensionality
- robustness across a wide range of problem classes
- provision of multiple good solutions
- ability to locate the region of the global optimum solution

Research on evolutionary computation in engineering design has a relatively long history. It was initiated in Europe in the early seventies by Rechenberg [31] in the areas of fluid mechanics, pipe design and structural engineering. Early applications of EC in structural engineering [32,33] used ES which evolved from structural optimization approaches in the early 1960’s. Further significant progress in this area has taken place mainly during the last fifteen years. In the United States, Goldberg [34,35] did the first application of GAs, which emerged from the machine learning community, in engineering optimization in the area of complex gas pipeline systems. Just about the same time, in the late 80’s and early 90’s, many researchers started applying this new optimization method to a large spectrum of engineering design problems. Current state-of-the-art reviews are provided in [6,30,36-45].

3.1 Creative Design

Evolutionary design optimization and creative evolutionary design are the two categories of evolutionary design that are particularly relevant to civil and structural engineering applications. From a computational point of view, the dividing line between the two categories is not sharp and is mostly related to the potential for achieving novelty/creativity during the processes of generating design concepts as well as properties that novel/creative designs need to possess. For Gero [46] creativity in design "is not simply concerned with the introduction of something new into a design, although that appears to be a necessary condition for any process that claims to be labeled creative. Rather, the introduction of 'something new' should lead to a result that is unexpected (as well as being valuable)." Gero concludes that an evolutionary design process is creative when it explores not only values of attributes (decision variables) within individual design spaces but also evolves the number of these attributes, i.e. when changes in the representation space occur. Similarly, Boden [47] suggests that achieving creativity is only possible by going beyond the bounds of a representation, and by finding a design that could not have been defined by that representation. The same concept was explored by Arciszewski and co-workers in the context of Inferential Design Theory [48] and constructive induction [49]. A detailed discussion of commonly used representations in evolutionary design, including generative representations supporting creative design processes can be found in section 4.

Less restrictive definition of creativity in design was given by Rosenman [50]. He suggested that the distinguishing feature of all creative evolutionary design systems is the ability to generate entirely new designs starting from little or almost no knowledge (for example when starting with random initial conditions), and being guided throughout the evolutionary process only by performance criteria.

3.2 Evolutionary Design and the Theory of Inventive Problem Solving (TRIZ)

Creativity in evolutionary design can also be analyzed from a broader perspective, namely based on the theory of inventive problem solving (TRIZ) introduced by Altshuller [51,52]. Altshuller discovered that the evolution of engineering systems is not a random process, but it is governed by a class of paradigms. These paradigms can be subsequently used to develop a system considering its technical evolution, i.e. by determining and implementing innovations. Altshuller introduced five levels of innovation in the context of an engineering design problem [49]:

A. Selection

"A design concept is selected from a group/class of known concepts in a given engineering domain."

This level of innovation corresponds to **an EA using only selection operation** and that is initialized with a population of known design solutions, rather than randomly generated ones.

B. Modification

"A design concept is produced as a combination and/or modification of known design concepts from a given domain. The modification process can be performed either deterministically or using a random generation process."

This paradigm is equivalent to **an EA searching for an optimal solution in a parameterized representation space** of a class of engineering designs. Rosenman's definition of creativity in design is most closely related to this paradigm and hence it becomes obvious that his prerequisites of creativity are fairly weak when compared to Altshuller's innovation taxonomy.

C. Innovation

"A design concept is produced as a combination of known concepts from a given domain and other domains."

This paradigm can be best represented as the **island model EA** where various populations of designs evolve independently and occasionally exchange some individuals through a migration process. The migrations can model injection of knowledge from other domains to a particular engineering domain.

D. Invention

"A design concept is produced as a combination of known concepts from a given domain and new concepts based on a new technology, which have been recently introduced."

EA can achieve this level when it evolves not only the values of attributes but also the attributes themselves [53]. In other words, it can use various transformation operators [49] for a representation space including attribute addition (introduce new attributes/genes to the representation space), attribute elimination (removing unimportant attributes), attribute abstraction (combining attributes into larger units, or components, and subsequently exploring the component based representation [54]), and attribute construction (creating new attributes by a simple or complex transformation of the initial attributes). This level of innovation is most closely related to

Gero's definition of creativity in design as well as changes in the representation space introduced in the constructive induction process [49].

E. Discovery

"A design concept is produced as a combination of known concepts from a given domain and new concepts based on new scientific principles."

This highest level of innovation in Altshuller's taxonomy can most likely be achieved by evolutionary design processes. However, special types of representations, namely the generative representations [55] (described in section 4), seem to be necessary to accomplish it. Generative representations use compact representations (genotypes) of existing design knowledge and mappings that translate these representations to actual designs (phenotypes). The mappings can reuse elements of the representations during the process of translation. Thus, the compact representations can be thought of as storing existing knowledge on a given engineering domain, whereas mappings correspond to new scientific principles that can transform the known concepts to new, and possibly creative, design concepts. The mappings are usually simple programs that take the compact representations as input and produce the actual design concepts as output. Despite their simplicity, they can generate designs that can be defined as creative [56]. Recently, Wolfram [57] suggested that all scientific principles and natural processes can be modeled in terms of simple programs that can nevertheless produce complex behavior. **EAs using generative representations** will search both the space of compact representations and the space of simple transformation programs (scientific principles) and will generate creative design concepts.

The first two paradigms, i.e. selection and modification, can only produce *routine designs*. In both cases, no changes occur in the representation space [49]. The last three paradigms, i.e. innovation, invention and discovery, can generate *novel/creative designs*. In all these cases, changes in the representation space do occur [49].

3.3 Emergence

Emergence is an important property which is closely related to creativity in design. Gero [58] defines emergence as "a process of making features explicit, that were previously only implicit." He also suggests that emergence plays an important role in introducing new attributes to the representation space [46]. Emergence can also be easily recognized through the visual examination of representations of structures, for example of structural patterns of steel structural systems in tall buildings [59].

The notion of an *emergent concept* generation has also been introduced by Arciszewski et al. [49] as a part of a constructive induction process that was originally proposed in the field of machine learning. An *emergent design concept* is defined as a constructed attribute (representing an unknown design concept) whose introduction may simplify and improve effectiveness or quality of a design process. A constructed attribute is derived from the initial attributes by an application of constructive induction operators. It is usually more abstract than the attributes from which it was derived.

3.4 Integrated Design

Most applications of evolutionary methods in civil and structural engineering were focused on a detailed design stage of a design process, where the objective was to find the optimal configuration of attribute values for a previously selected and parameterized design concept. Thus, only routine design concepts could be generated, even though they were optimized with respect to some objective. An overview of the SOTA in evolutionary design applications in civil and structural engineering can be found in section 8.

There has also been some work in applying evolutionary design methods at the conceptual stage of an engineering design process, where the emphasis is put on the generation of novel and original design concepts, and not on finding the globally best solution in terms of numerical values in the context of a specific design concept. Gero and Schnier [60] worked on the evolution of a design knowledge representation, using genetic algorithms and Rosenman and Gero [53] used genetic engineering to evolve architectural floor plans. Arciszewski et al. [61] used evolutionary computation to produce creative designs. Bentley [62] developed a generic evolutionary design system, which was able to evolve a range of various designs from scratch. The system performed evolutionary design with an emphasis on the evolution of creative design concepts rather than their optimization.

The concept of integrated design utilizing various forms of evolutionary computation at each stage of a design process as well as incorporating designer's knowledge and intuition within the search and exploration process has been pioneered by Parmee [42,63]. In the mid-90's, research was initiated on the utility of evolutionary/adaptive search within the generic domain of an engineering design process as a whole. Parmee, following Pahl and Beitz [64], distinguishes three major stages of an engineering design process: *conceptual design*, *embodiment design*, and *detailed design*. He considers conceptual design as "a search across an ill-defined space of possible solutions using fuzzy objective functions and vague concepts of the structure of the final solution." Embodiment design operates with a selected (during the conceptual design stage) initial design configuration and aims to further specify the subsets forming the whole system. Design decisions at this stage are made based on both qualitative and quantitative criteria which usually are difficult to be formally defined using mathematical models and hence difficult to include in a scalar objective (fitness) function. Finally, at a detailed design stage, design decisions are made based on solely quantitative criteria which are well described by mathematical models, even though they may be computationally expensive and may require complex analysis techniques. Contrary to traditional and simplified definitions of engineering design process which assume little or no interaction between the stages [64], Parmee argues that considerable overlaps exist among the three stages and they should be taken into account in the integrated design model. He suggests that a model of a design optimization process should be considered to "represent a long-term, highly complex process commencing with high-risk conceptual/whole-system design and continuing through the uncertainties of embodiment/preliminary design to the more deterministic, relatively low-risk stages of detailed design and the eventual realization of an optimal engineering solution."

The objective of Parmee's integrated design was to develop co-operative frameworks involving a number of evolutionary/adaptive computing techniques and integrate them with each stage of the engineering design process. During this research, various forms of evolutionary computation were considered in the context of integrated design, including Structured Genetic Algorithms [16], GAANT Algorithms [65], and Ant Colony Algorithms [66,67] as well as constraint satisfaction [68]. Next, Parmee investigated evolutionary computation in the context of searching "whole-system design hierarchy" described by both nominal and numerical attributes [69], and he applied it to designing hydropower systems [70]. Later, Vekeria and Parmee [71] proposed the use of evolutionary computation in conceptual design of structural systems, including the determination of the topology of their members. Recently, he has been focused on the "innovative conceptual design" in the context of variable mutation cluster-oriented Genetic Algorithms (vmCOGAs) and successfully used them in the area of aerospace engineering [72].

4. Evolutionary Design Representations

Representations in engineering design incorporate both representation of an *artifact* being designed as well as representation of a *design process,* i.e. a process by which the design is completed. The line distinguishing artifact representation and design process representation is often blurred. Building a representation of an artifact is similar to the process of its numerical/mathematical modeling in engineering science. It is, however, significantly broader because it encompasses much more knowledge than can be set into mathematical formulas and their numerical realizations. Generally, a representation of a designed artifact should describe its function, form, intent, legal requirements, etc. Advances in computer science, and evolutionary computation in particular, made it possible to use symbolic representations to describe objects, attributes, relationships, concepts, etc. Thus, it is now possible to capture more abstract and conceptual design knowledge [73].

A *representation* of an engineering design is as a computational description of an engineering system (that usually does not yet exist) expressed in terms of attributes [49]. In the most straightforward EC representation, each gene corresponds to an attribute and represents a dimension of the search space. Each such dimension can have an appropriate set of values (discrete or continuous) that a feature represented by this dimension can take on. In the simplest case, these representations use binary genes denoting the presence, or absence, of a feature. In such representations each individual consists of a fixed-length binary string of genes, or a genotype, representing some subset of a given set of features. Often, in complex engineering applications, multi-valued attributes are more natural to use [6].

A *representation space* for an engineering design is a multidimensional space spanned over attributes that are used to describe an engineering design [49]. Attributes can be *symbolic* (when they take values from an unordered or partially ordered set) or *numerical* (when they take numerical values representing quantities or measurements). Symbolic attributes that take values from an unordered set are called *nominal* attributes; when they take values from a partially ordered set, they are called *structured*. Design concepts are typically described in terms of symbolic attributes. Numerical attributes are used for a detailed description of a design.

A *design concept* is understood as a description of a future engineering system, actual or abstract, in terms of a feasible combination of symbolic attributes and their values. After a conceptual design process is completed, a given design concept is used next in the detailed design process to produce a detailed design. A *detailed design* is understood here as a detailed description of a future engineering system in terms of both symbolic and numerical attributes (dimensions, weights, etc.) [49].

4.1 Optimality vs. Creativity

A choice of a particular representation of an engineering system for an evolutionary design process is highly influenced by the designer's goal, i.e. whether the emphasis is on optimality in terms of numerical values in the context of a specific design concept, or on generation of creative design concepts. When the focus is on finding an optimal design, designers' attention is usually restricted to a particular concept or at most several concepts of existing designs. In this case, design representations usually take a form of parameterizations of an engineering system, or its parts. The parameters are then encoded as genes and their alleles are evolved using evolutionary algorithms in order to find the best design that maximizes (or minimizes) given objective(s). Thus, for strictly engineering optimization problems, representations should be *direct* (i.e. they should encode possible solutions) and *parameterized* (allowing only for slight variations). Traditional representations frequently used in engineering optimizations problems, like binary representations, integer representations, and real-valued representations can be included in this category. Additionally, representations used in optimization problems usually incorporate domain knowledge, to smaller or larger extent, in order to make the search more efficient.

Creative evolutionary design requires, however, more general and usually more complex representations. Representations that have been used in creative design are diverse but nevertheless share some similarities. Typically, phenotype representations are quite general and thus capable of representing large numbers of alternative shapes, forms, or morphologies (forms together with structures) [29]. They range from direct representations, as in voxel-based representations [74] or array-based representations [75,76], to highly *indirect* representations, i.e. representations that do not encode solutions but rather rules on how to build these solutions. The most popular examples of indirect representations are grammars [77], trees [78,79], shape grammars [80-83], graphs and matroids [84], cellular automata [85,86], L-systems [55,87,88], and embryogenies [56].

4.2 Selecting Appropriate Design Representations

Gen and Cheng [41] discuss five major requirements for designing good representations (genotype-phenotype mappings) for evolutionary design problems:

A. "Non-redundancy"

"The mapping between encodings and solutions should be 1-to-1."
There should be a unique pairing of each element of a genotypic space with a corresponding element of a phenotypic space. Out of all three possible cases, the *1-to-n* mapping should be particularly avoided because it corresponds to multiple phenotypic representations of the same genome. In this case, an additional procedure would have to be employed to determine the actual phenotype.

B. "Legality"

"Any permutation or combination of an encoding corresponds to a solution."
It is important to distinguish between two basic concepts: *infeasibility* of a solution and its *illegality*. Infeasible solution means that a phenotype decoded from a genotype lies outside of a feasible region (defined by the constraints) in the phenotypic space. Illegal solution means that a genotype does not represent any phenotype for a given problem. The implicit significance of the legality requirement is that it implies that standard genetic operators can be easily applied to a representation satisfying this requirement.

C. "Completeness"

"Any solution has a corresponding encoding."
This requirement guarantees that any phenotype has a corresponding genotype, and hence it is accessible to genetic search.

D. "Lamarckian property"

"The meaning of alleles for a gene is not context dependent."
This requirement "concerns the issue of whether or not one chromosome can pass on its merits [learned traits] to future populations through common genetic operators" [89]. If the **meaning** of alleles for a gene is interpreted in a context-dependent manner, as in the *non-Lamarckian* case,

the offspring usually inherit nothing from parents. Generally, the representation should have the Lamarckian property so that offspring can inherit goodness from parents.

E. "Causality" (also known as Continuity)

"Small variations on the genotype space due to mutation imply small variations in the phenotype space."

This requirement focuses on the preservation of neighborhood structures. The appropriate choice of genotype-phenotype mapping in combination with the genetic operators is important for a successful evolutionary search process [90]. For a successful introduction of new information by an operator, the operator should preserve the neighborhood structure in the corresponding phenotype space. Search processes that preserve the neighborhood structure are said to exhibit *strong causality*.

4.3 Taxonomy of Representations

Representations used in evolutionary design have been classified with respect to many different criteria. Table 2 presents a compilation of classification schemes in which attributes and their values correspond to various categorizations of evolutionary design representations proposed by several researchers [1,55,91].

Table 2: A classification of EA representations.

No.	Attribute	Attribute Values	
1	EA level	Genotypic	Phenotypic
2	Structure	Linear	Nonlinear
3	Length	Fixed	Variable
4	Change during evolution	Static	Dynamic
5	Encoding scheme	Direct	Indirect
6	Accuracy of solution specification	Parameterization	Open-ended
7	Ability to reuse encoding	Non-generative	Generative
8	Genotype-phenotype correspondence	Explicit	Implicit

One of the most important representational issues is the choice between a *genotypic* and *phenotypic* representation. This issue has some important consequences not only for EC in general but also for evolutionary design. When one decides to use a genotypic representation (as it is the case in the canonical GAs) then an appropriate genotype-phenotype mapping has to be constructed, hopefully satisfying all five major requirements stated earlier. A particular attention has to be paid to satisfy the causality requirement. The lack of correlation between variation at the genotype level and variation at the phenotype level can cause serious problems [1]. When a genotypic representation is used, mutation and recombination operate at the genotypic level while the fitness evaluation and selection are performed at the phenotypic level. One of the advantages of using genotypic representations is the ability to reuse standard genetic operators for multiple problem domains.

Alternatively, one can just use phenotype level encodings (as it is the case in the canonical ES) to both explore and exploit a design space. The significant advantage of this approach is that no mapping between genotype and phenotype is necessary and hence all five requirements stated earlier are automatically satisfied. One can focus on achieving useful exploration only at the phenotypic level. The disadvantage of phenotypic representations is that the genetic operators become problem dependent and have to be carefully crafted for each individual problem domain [1]. Phenotypic encodings have been widely used within the ES community and applied to many engineering optimization problems.

A structure of an evolutionary design encoding is another relevant criterion. Generally, representations can be divided into *linear* and *nonlinear*. A linear representation can be thought of as a 1-dimensional representation usually in a form of a string (binary, real-valued, integer-valued), list, etc. Nonlinear representations, on the other hand, have 2-, or higher- dimensional structure, e.g. trees, arrays, etc.

Another distinguishing property of evolutionary design representations is their length. They can be divided into two groups: *fixed-length* and *variable-length* representations. The length of a genome is constant during an entire evolutionary process when fixed-length encodings are used. It is not the case with variable-length representations where an individual can be represented by a genome that changes its length every generation. Consequently, a population may consist of individuals whose genomes have different lengths. Fixed-length representations have been widely used in evolutionary design optimization while variable-length representations have been applied to creative evolutionary design [36].

Depending on whether, or not, the representation can change during an evolutionary design process, one can divide representations into *static* and *dynamic*. This is a more general classification than the one based on a change of the length of a genome because it considers not only a time-dependent change of the length of a genome but also time-dependent changes made to its structure.

Direct representations encode essentially actual design concepts, while *indirect* representations encode rules on how to construct these concepts. Again, direct representations are used mostly for evolutionary design optimization and indirect encodings for evolving creative design concepts [55].

In the case when the topology of a design is established in advance and specified in sufficient detail, i.e. it is parameterized; the representation is called a *parameterization*. On the other hand, when the topology of a design is changeable then the representation is called *open-ended*.

Representations that can reuse some parts of an encoded design during the phenotype construction phase from a genotype are called *generative*. Generative representations are always indirect. *Non-generative* representations can not reuse elements of the encoding. They can be either direct or indirect. Generative representations offer several advantages when compared to non-generative ones. Their ability to reuse elements of an encoded design improves the search efficiency in large design spaces as well as scalability by capturing design dependencies [55].

Depending on the nature of the relationship between the elements of a genotype and the elements of the generated phenotype, generative representations can be further classified as *implicit* or *explicit*. Implicit representations consist of a set of simple rules (e.g. cellular automata) that implicitly specify a design property, e.g. its shape, through an iterative construction process. Explicit representations are like procedural programs for constructing designs in an explicit manner.

Recently, there have been several attempts to coevolve representations of engineering systems during the evolutionary process. This corresponds to the process in which a learning system adapts its own representation. De Jong and Oates [92] proposed a coevolutionary approach to representation development where building blocks and their assemblies are coevolved. Also, Gero and Schnier [60] worked on the evolution of the design knowledge representation, using genetic algorithms, in the context of case–based design. Such evolution is often necessary to produce inventive designs.

4.4 Traditional Design Representations

The majority of evolutionary design applications in structural engineering reported in the literature used relatively straightforward representations consisting of either binary strings or real-valued vectors. Thus, it is important to be aware of the strengths and weaknesses of both common approaches to represent engineering systems.

Binary representations are standard representations for canonical GAs. The most straightforward and at the same time most common approach involves binary strings of fixed length. This type of representation is best suited for problem domains where solutions can be naturally represented as binary vectors, e.g. in some combinatorial optimization problems. In engineering design this type of representations has been widely used in structural topology optimization, e.g. in the ground structure approach [93].

When a problem domain cannot be defined in terms of binary vectors, then a mapping from the binary space (genotypic space) to the domain space (phenotypic space) is necessary. Using this principle, binary string representations have been applied to continuous parameter optimization problems [94]. In this case, a mapping between binary strings and real-valued parameters had to be specified. This approach has been widely used in many engineering design applications. Its advantage is that the standard GA operators (e.g. the bit-flip mutation, and one-, or two-point crossover) can be used. There are, however, some important drawbacks of this approach, too. Michalewicz [94] argues that it is not appropriate because the problem space the GA is operating in is fundamentally different than that of the originally defined problem. Thus, search and optimization are conducted in a different space than the original one. Hence, the optimal results obtained in the binary search space might in fact not be optimal for the original problem. The genotype-phenotype mapping also introduces some additional nonlinearity to the objective function, and hence it may happen that the modified problem is more difficult to solve than the original one. Bäck [95] points out another serious drawback of mappings from continuous to binary spaces. The mappings impose some granularity (resolution) and hence not all the points in the original continuous space can be expressed as binary vectors. So, it is possible that the optimal solution will not be found simply because it is not represented in the binary search space.

Another important problem with binary representations is related to the fact that one of the five major requirements on genotype-phenotype mappings, namely causality or continuity requirement, does not hold. In other words small changes in the binary space correspond to large changes in the real-valued parameter values and vice-versa. A frequently employed solution in this case is to use Gray encoding scheme [95].

Real-valued representation spaces have been traditionally used by ES researchers to solve complex continuous parameter optimization problems. Historically, they have been applied to engineering design problems, specifically to various fine tuned optimization problems. In ES, real-valued representations have traditionally been used as phenotypic representations, where no mapping between a genotype and a phenotype is necessary. Thus, the drawbacks associated with the mappings are eliminated in this case. There are, however, two major problems with real-valued representations that are somehow related. First, real-valued encodings allow for representation of only very specific problem domains, and that usually corresponds to fine-tuned optimization problems. As such, they are not applicable for creative design problems as it was discussed earlier. The second problem is that not every design problem can be expressed as a real-valued vector. There are many design problems, conceptual design problems being a good example of, that involve some symbolic or qualitative variables which cannot be encoded as real-valued parameters.

As stated earlier, representations are one of the three key elements in a successful implementation of evolutionary design. Throughout the years, enormous amount of experimental work has been devoted to studying various types of evolutionary representations. Despite this fact, very little is known theoretically about their influence on the performance of an EA. Initial framework for evolutionary representation theory has been recently proposed by Rothlauf [96], but it is just the beginning of research on this important topic in EC.

5. Constraint-Handling Methods in Evolutionary Design

The vast majority of engineering design problems involves constraints of some kind. Thus, appropriate methods of handling constraints are extremely important for any optimization/search mechanism exploring designs spaces. Evolutionary algorithms, on the other hand, are unconstrained optimization procedures and hence it is necessary to somehow incorporate constraints into them. This section reviews the SOTA in constraint-handling methods in the context of evolutionary design. It also provides references to actual applications in structural engineering.

Coello Coello [97] classifies constraint-handling methods used with EAs into the following five major groups:

1. Penalty functions
2. Special representations and operators
3. Repair algorithms
4. Separation of objectives and constraints
5. Hybrid methods

5.1 Penalty Functions

Penalty functions have traditionally been the most common way of handling constraints incorporated in EAs [35,98]. This method was initially proposed in the early 1940's in the context of traditional mathematical optimization by Courant [99] and later extended by the operation research (OR) community in the 1960's [100,101]. In the 1980's, penalty functions have been adopted by EC researchers to solve constrained optimization problems [35,102] and since then have become the most popular, albeit not best as it has been shown in several studies [103], method of handling constraints. Penalty functions effectively transform a constrained design problem into an unconstrained one by augmenting the objective function with a penalty term whose value determines the amount of constraint violation present in a particular solution [97]. Contrary to classical optimization methods which use penalty functions of two kinds (i.e. exterior and interior), evolutionary design focused almost exclusively on exterior penalty functions because they do not require initial feasible solution to start with.

Various types of penalty functions have been proposed and studied. A general classification of the most commonly used types of penalty functions is presented below [97]:

1. *Static penalty functions* which remain constant during an entire evolutionary process [102,104].

2. *Dynamic penalty functions* which change throughout an evolutionary run (usually increase over time) [105].
3. *Annealing penalty functions* which use techniques based on simulated annealing [106].
4. *Adaptive penalty functions* which change according to feedback received from the search process [107-111].
5. *Coevolutionary penalty functions* in which solutions are evolved in one population and penalty factors evolve in another population [112].
6. *Death penalty functions* which immediately reject infeasible solutions [113].

One of the major challenges in any application of penalty functions concerns achieving an appropriate balance of the penalty value. Large penalty values discourage EAs from exploring infeasible regions and the search is quickly moved inside the feasible region. On the other hand, low penalty values do not prohibit EAs from searching infeasible regions most of the time. As a result of these findings, several EC researchers proposed the 'minimum penalty rule' which states that "penalty should be kept as low as possible, just above the limit below which infeasible solutions are optimal" [97]. The problem with this formulation, especially for structural design applications, is that usually the constraints are not expressed in an algebraic form but instead as outcomes produced by structural analysis packages. Hence, an exact location of the boundaries between feasible and infeasible regions cannot be specified.

Methods of designing/configuring penalty functions for EC applications have been studied in Richardson et al. [103]. They offer several guidelines/heuristics that can be used to make evolutionary search in constrained design spaces more efficient:

- "Penalties which are functions of the distance from the feasible region are better than those which are merely functions of the number of violated constraints.
- For a problem having few constraints, and few solutions, penalties which are solely functions of the number of violated constraints are not likely to find solutions
- Good penalty functions can be constructed from two quantities, the maximum distance and the expected distance to the feasible region.
- Penalties should be close to the expected distance to the feasible region, but should not frequently fall below it. The more accurate the penalty, the better the solutions will be found. When penalty often underestimates this distance, then the search may not find a solution."

A number of applications showed, however, that there are many difficulties associated with penalty functions [103], including, for example, a problem of defining good penalty factors. Thus, over the years, alternative approaches to handling constraints have been proposed by EC researchers.

5.2 Other Methods

Alternative attempts to handle constraints in evolutionary design include the development of *special representations* that simplify the shape of the search space and *special genetic operators* that preserve feasibility of generated solutions during the evolutionary run. Examples of applications of these methods include Bean's [114] 'random keys encodings', Davidor's [115] 'analogous crossover,' Michalewicz's [94] GENOCOP, and Kowalczyk's [116] constraint consistent GAs. Schoenauer and Michalewicz [117] proposed a method that restricts the search to the boundary of a feasible region. It is based on a heuristic that in many cases the global solution lies on the boundary of a feasible region. In this method, the search mechanism crosses the feasibility boundary back and forth and special genetic operators are used to restrict the variation to the boundary of the feasible region [118]. The last set of methods in this category uses decoders [119]. In this case, chromosomes encode instructions on how to construct feasible solutions [120]. Each decoder imposes a mapping between a feasible solution and a decoded solution [120,121]. Koziel and Michalewicz [120] reported that decoders provided much better results than any other constraint-handling method on a representative set of test problems. They seem to be a very promising area of research in structural design because they can be used with problems of any dimensionality and do not require the objective function given in an algebraic form [97].

Repair algorithms are particularly well-suited for combinatorial optimization problems [122]. They are particularly efficient when the cost of transformation of an infeasible solution into a feasible one is low [97]. They have been applied to many optimization problems [123-126]. An interesting aspect of repair algorithms is whether, or not, a repaired individual should replace the original infeasible individual in the population. The spectrum of possible choices ranges from no replacement (repaired individuals are used only for evaluation and the original individuals remain in the population) [123,127] to the full replacement (all infeasible individuals are replaced with the repaired ones) [128]. Also, some intermediate approaches have been suggested where original infeasible solutions are replaced with some probability by the repaired solutions [129]. In structural design, repair algorithms have been used e.g. in [130] to repair design concepts of steel structural systems in tall buildings not satisfying the symmetry requirement.

Another group of constraint-handling techniques can be broadly categorized as methods based on *separation of constraints and objectives* [97]. Most representative techniques in this category include:

1. *Competitive coevolution* in which potential solutions (possibly infeasible) are evolved in one population and constraints are contained (but not evolved) in another population [131]. Individuals representing potential solutions have high fitness when they satisfy a large number of constraints from the other population. On the other hand, an individual representing a constraint has high fitness if this constraint is violated by many potential solutions.
2. *Superiority of feasible points* which assumes that all feasible solutions are better than infeasible ones [132,133].
3. *Behavioral memory* that uses a special technique of ordering constraints in which the algorithm proceeds by sequentially satisfying the constraints imposed on the problem [134].
4. *Multiobjective optimization methods* in which an original single-objective problem is transformed into a multiobjective one by treating all constraints in the original problem as objectives in the transformed problem [135-139]. A discussion on evolutionary multiobjective techniques is presented in section 6.

Finally, the last category of constraint-handling methods includes *hybrid methods* in which EAs are combined with other methods to solve constrained problems. In this category, several interesting methods were proposed, including:

1. *Lagrangian multipliers* in which a hybrid EA is formed by integration of a penalty function method with mathematical programming methods including the primal-dual method and augmented Lagrangian function [140] that guarantees the generation of feasible solutions during the search [141,142].

2. *Fuzzy logic* in which an EA is combined with fuzzy logic. In this method the original constraints are replaced by fuzzy constraints to allow a higher degree of tolerance for violating constraints that may occur close to the boundary of the feasible region [143,144].

3. *Immune system* models which have been initially proposed to maintain diversity in multi-modal optimization problems [145,146] and later extended to solve constrained optimization problems [147-149].

4. *Cultural algorithms* which have been initially used to model cultural evolution [150] and later applied to numerical optimization problems involving constraints [151,152].

5. *Ant colony algorithms* inspired by colonies of real ants and initially proposed for solving combinatorial optimization problems [67,153] and subsequently extended to constrained optimization problems [66,154].

Excellent state-of-the-art reviews presenting theoretical and practical aspects of constraint-handling methods in evolutionary computation can be found in [38,97,98,155,156].

6. Multiobjective Evolutionary Design

Evolutionary multiobjective optimization (EMOO) is one of the most active research subfields within the EC community nowadays. EMOO methods are also highly relevant to engineering design problems because they were designed to handle multiple conflicting objectives which usually occur in real-world design problems. This section introduces the SOTA in evolutionary multiobjective optimization and presents recent developments in applications of these techniques to structural design problems.

There are two major goals of multiobjective optimization. First, one wants to find a large number of Pareto-optimal [157] solutions to a given problem. Second, the solutions to the problem should be widely differentiated [158]. Classical search and optimization methods (like weighted sum method [159] or ε-constraint method [160]) are not efficient for multiobjective problems because most of them cannot find multiple solutions in a single run, and even multiple runs do not guarantee finding different optimal solutions. On the other hand, EAs are well-suited to solve these kinds of problems because they are population-based and this property allows them to find an entire set of Pareto-optimal solutions in a single run. Additionally, they are significantly more robust, compared to the classical methods, particularly when issues like the shape or continuity of the Pareto front are a matter of concern [161].

Initial research on using evolutionary methods for solving multiobjective problems was conducted by Rosenberg [162]. He suggested, but did not implement, a genetic search method involving multiple biochemical properties and objectives of a population of single-celled organisms. The first actual implementation was conducted by Schaffer [163]. In his dissertation, he proposed and successfully applied the vector evaluated genetic algorithm (VEGA) to multiclass pattern discrimination tasks in machine learning. Next significant progress in the field came with Goldberg's [35] non-dominated sorting procedure. Since that time, many researchers have developed various versions of multiobjective optimization algorithms. The most popular approaches reported in the literature include [158,161,164]:

1. *Aggregating functions* in which multiple objectives are combined into a single one using addition, multiplication, or any other combination of arithmetic operations [165]. Frequently, the weighted sum approach is adopted in which the objectives are multiplied by weighting coefficients representing the relative importance of the objectives [166,167]. The major drawbacks of this method include difficulties in determining the appropriate weights and the fact that improper Pareto solutions may be generated in the presence of non-convex search spaces regardless of the weights used [161].
2. *Vector evaluated genetic algorithm (VEGA)* proposed by Schaffer [168]. It handles multiple objectives by modifying the survival selection mechanism of the simple GA. Several variations of the original VEGA have been proposed and applied to various problems, including a groundwater pollution containment problem [169], and conceptual design of airframes [170].
3. *Target vector approaches* in which targets or goals have to be defined by a decision maker for each objective [161]. This group of approaches includes goal programming [171], goal attainment [172], and min-max approach. This last method, the weighted min-max, has been used by Haleja and Lin [173] to optimize a 10-bar plane truss in which weight and displacement were to be minimized, and by Coello Coello and Christiansen to optimize I-beams [174] and truss designs [175].
4. *Multi objective genetic algorithm (MOGA)* proposed by Fonseca and Fleming [176]. It defines a rank of an individual based on the number of individuals in the current population by which it is dominated. MOGA has been used in many engineering design applications including for example a gas turbine controller [177] and supersonic wings [178,179]. Grierson and Khajehpour applied a variation of MOGA (called MGA) to conceptual design of office buildings [180].
5. *Non-dominated sorting genetic algorithm (NSGA)* defined by Srinivas and Deb [181] and based on Goldberg's [35] notion of non-dominated sorting with a niche and speciation method. An improved version of this algorithm, called NSGA-II [182], equipped with elitisms and parameter-free sharing approach has been recently applied to a topological optimum design problem by Hamda et al. [183]. In their approach, both the mass and the maximum displacement of a cantilever plate were minimized. Deb and Goel [184] used a hybrid approach, NSGA-II and a hill climber, to solve several engineering shape optimization problems.
6. *Niched Pareto genetic algorithm (NPGA)* proposed by Horn and Nafpliotis [185]. It uses a tournament selection scheme based on Pareto dominance.
7. *Strength Pareto evolutionary algorithms (SPEA)* proposed by Zitzler and Thiele [186] which integrates ideas from various existing evolutionary multiobjective optimization methods and adds some new elements to the evolutionary multiobjective algorithm.

Comprehensive surveys of various evolutionary multiobjective optimization methods, including detailed discussion on their strengths and weaknesses, can be found in [40,156,158,161,164,187].

7. Coevolutionary Design

Another important branch in evolutionary computation research that has recently received significant research attention is coevolution. The authors refer to coevolution as a phenomenon occurring when two or more populations (some researchers also include in this category single population models) simultaneously evolve and where no objective fitness function exists but rather individual's fitness is a subjective function of its interactions with individuals from coevolving populations [188,189]. Biological coevolution encountered in many natural processes has been an inspiration for a class of coevolutionary algorithms. Initial ideas of modeling coevolutionary behavior were formulated by Maynard Smith [190] and Axelrod [191,192]. The competitive approach to coevolution has been since widely used in many game-theoretic models that arise in various disciplines, including economics, decision sciences, social sciences, etc. Initial ideas were further extended by Hillis [193], Paredis [131,194], and others and resulted in a new optimization procedure called a coevolutionary genetic algorithm (CGA). Competitive coevolutionary models are especially suitable for problem domains where it is difficult to explicitly formulate an objective fitness function, for example in AI game-playing strategies, etc. Paredis [131] applied competitive coevolutionary algorithms to constrained optimization problems. Recently, they have been used e.g. to coevolve cellular automata and the training cases for the majority classification problem [195].

Potter and De Jong [196] proposed another approach to coevolution, namely a cooperative coevolutionary model. The motivation for this model comes from problem domains where explicit notions of modularity have to be introduced [197]. This model also provides appropriate framework for evolving solutions in the form of co-adapted subcomponents, and hence is of crucial importance for many engineering design problems. Usually, complex engineering design problems are decomposed into simpler problems and solved independently. This works fine for problem domains where the principle of superposition can be applied, i.e. for problems that can be linearly decomposed. That is no longer the case, however, for complex designs where nonlinear interactions take place among the subcomponents and make interacting members highly dependent on one another. For these domains cooperative coevolutionary model is more suitable because it allows for an explicit subcomponent co-adaptation. Potter and De Jong [27] proposed a cooperative coevolutionary architecture for evolving co-adapted subcomponents and defined a cooperative coevolutionary evolutionary algorithm (CCEA). This architecture has been subsequently analyzed from the evolutionary dynamics perspective [188,198] as well as from the perspective of collaboration methods that have been used [188,199].

In general, coevolutionary design processes can be defined by 7 major attributes shown in Table 3. They describe ways in which coevolutionary systems can be set up [188]. The attributes include the payoff quality, methods of fitness assignment, methods of interaction, update timing, problem decomposition, spatial topology, and population structure.

Table 3: Coevolutionary architectures described in terms of attributes [188]

No.	**Attribute**	**Attribute Values**		
1	Payoff quality	Cooperative	Competitive	Non-competitive
2	Methods of fitness assignment	Implicit	Explicit	
3	Methods of interaction	Sample size	Selective bias	Credit assignment
4	Update timing	Sequential	Parallel	
5	Problem decomposition	Partitioning methods	Temporal decomposition	
6	Spatial topology	Spatial embedding	Non-spatial embedding	
7	Population structure	Single	Multiple	

Coevolutionary models have been applied to several engineering design problems, particularly in architectural design. Maher and Poon [200] suggested that it is often the case in a design process that requirements are reconsidered when a design solution is offered. Maher [201] introduced the idea of coevolutionary design, where requirements and solutions evolve separately. Maher and co-workers [200,202-207] have been working on coevolutionary design in which two interrelated evolutionary processes occur. The first one is the evolution of design solutions while the second one is the evolution of requirements. In this case, the fitness function evolves with the requirements and it is different (local) at various stages of the coevolutionary design process. Also, the fitness function is used to identify the surviving solutions, but its convergence simply means that there is no progress in the evolution since no new and better solutions are being produced.

The only work known to the authors which uses cooperative coevolutionary algorithms in structural optimization was conducted by Nair and Keane [208]. They used CCEAs to optimize cross-sections of members of planar truss systems (single objective weight minimization problem). The optimized truss systems were decomposed and coevolved in separate populations.

8. Evolutionary Computation in Structural Engineering

The history of evolutionary computation in structural engineering can be traced back to the mid 1970's and early 1980's [32,33,102]. The vast majority of, if not all, early papers discussing EC applied to structural engineering were focused on structural optimization problems. Strong emphasis on various aspects of structural optimization remained the major focus of research in this field until now with relatively few exceptions which mostly addressed the issues of creativity in structural design and more sophisticated ways of representing structural systems [209].

Emergence of EC in structural optimization was a consequence of encountered problems and deficiencies of formal methods, including mathematical programming and the optimality criteria method [210], when applied to more complicated structural design domains. Formal structural optimization methods based on the assumption of continuity worked well on relatively well-formed problems in which the structural configuration of members was assumed and fixed during an optimization process while the task was to find the optimal sizing (dimensions) of members' satisfying at the same time imposed design requirements and constraints. The simple generalization of this problem by allowing variations of a system's configuration greatly increased the complexity of the optimization task and rendered many traditional methods inadequate. This issue became a starting point for a development of two major approaches to structural optimization that exist today: enhanced formal methods and heuristic methods.

8.1 Structural Design Problems

The problems addressed by structural optimization can be divided into three major categories:

- Topology (layout) Optimization also known as Topological Optimum Design (TOD) – looking for an optimal material layout of an engineering system
- Shape Optimization (SO) – seeking optimal contour, or shape, of a structural system whose topology is fixed
- Sizing Optimization – searching for optimal cross-sections, or dimensions, of elements of a structural system whose topology and shape is fixed

A structural design problem in each of the categories can be further classified as a continuum or discrete optimization problem. Figure 1, a modified version of a figure presented in [211], shows the three categories of structural optimization for continuum design problems while Figure 2 shows the same categories for discrete problems. The three categories are closely related to three major stages of engineering design process described earlier, i.e. TOD is conducted in the conceptual design stage, SO in the embodiment design stage, and finally sizing optimization is performed in the detailed design stage. As stated earlier, the three categories of structural optimization problems have been addressed by both formal optimization methods and heuristic methods.

Formal methods have been most successful when applied to sizing optimization problems which are usually well-defined in terms of mathematical models. Mathematical programming methods [212] and optimality criteria method [210] have been efficiently applied to solve these problems. Heuristic methods, including GAs, have also been applied to structural sizing problems [213,214]. On the other hand, TOD problems, located on the other end of the structural complexity spectrum, have been most successfully approached using heuristic methods, including simulated annealing [215] and EAs [76,216-218]. Structural shape optimization has been a kind of middle ground where both formal and heuristic methods are used and complement one another.

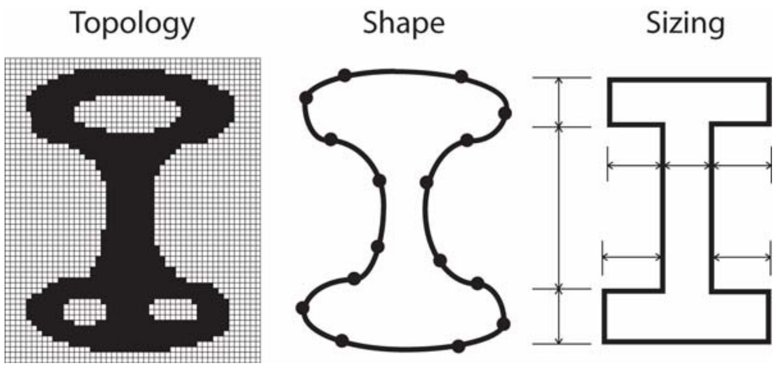

Figure 1: Topology, shape, and sizing optimization for continuum structural design problems

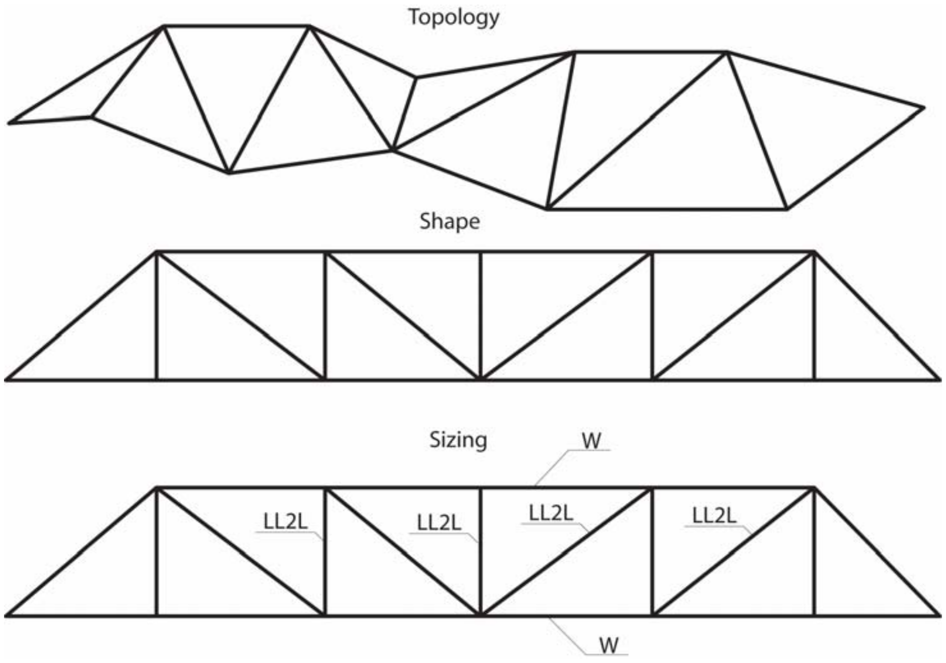

Figure 2: Topology, shape, and sizing optimization for discrete structural design problems

8.2 Topological Optimum Design

TOD has been an area of significant research efforts for the last forty years. Initial investigations in the late 1970's and early 1980's were conducted using formal methods. Generally, TOD problems can be divided into two major groups: *continuum TOD* and *discrete TOD*. In the continuum TOD, the design domain is discretized into small, rectangular elements (rectangular grid) where each element contains material or void. Formal methods addressing this problem include the homogenization method [219] in which each element in a grid contains composite material of continuously-variable density in [0,1] and orientation. Xie and Steven [220] proposed Evolutionary Structural Optimization (ESO) method which follows the concept of removing lightly stressed elements. The name of this method is confusing because the method is not based on EC principles but rather evolution is understood in a more general context as a process of gradual removal of inefficient material from a structure. The EC approach to the continuum TOD problem based on GAs has been developed by Sandgren et al. [221] and Jensen [217]. In their approach, a GA determines the optimal layout of material and void in a cantilever plate (represented as a bit array) such that the structure's weight is minimized subject to displacement and/or stress constraints. This work has been subsequently extended by Chapman et al. [218] to optimize finely-discretized design domains and to obtain families of highly fit designs. Recently, more advanced forms of representations for continuum TOD problems have been proposed, including Voronoi-based representations [222,223], which are based on concepts of Voronoi diagrams studied in computational geometry, and IFS representations based on fractal theory [209]. Also, Hamda et al. [183] considered a continuum TOD as an evolutionary multiobjective optimization problem.

Discrete TOD problems consist in determining the optimal element connectivity from a finite, albeit large, number of possible connections [224]. Two major problem domains addressed in early research in this area include truss structures and frame structures. An initial problem formulation in the context of linear programming using the ground structure approach was proposed by Dorn et al. [93]. While traditional linear programming methods proved to be successful in finding optimal topologies for small problems, they were rendered inadequate when the size of the problems considered was scaled up (increase in the number of design variables or the number of grid points in the ground structure approach). The discontinuous nature of this design problem was another reason for inefficiency of formal methods. Initial applications of GAs to optimize topology of discrete-member trusses were conducted by Shankar and Hajela [225], Hajela et al. [226], Grierson and Pak [227], and Hajela and Lee [216]. Bramlette and Bouchard [228] used EC to three-dimensional structures in the context of aircraft design. Koumousis & Georgiou [229] applied GAs to the topological optimization of steel truss structures. Bohnenberger et al. [230] applied GAs to optimize topologies of truss structures in pylons. Rajan [231] applied GAs to optimize topology, shape and member sizing of truss structures. Nakanishi and Nakagiri [232,233] used GAs to solve 2D topology optimization problems for both frames and panel structures. Rajeev and Krishnamoorthy [234] used variable-length string representations to optimize truss structures. Murawski et al. [235] and Kicinger et al. [130] applied ES to optimize topology of steel structural systems in tall buildings. Soh and Yang [236] introduced GP-based approach to TOD of truss structures. In a subsequent work [237], they proposed a GP-based methodology for the automated optimum design of structures. Recently, Azid et al. [238] applied a GA with real-valued representations to optimize topologies of three-dimensional trusses.

SOTA reviews of current research in formal methods for TOD problems can be found in Rozvany et al. [239], Bendsoe and Sigmund [240], and Xie and Steven [241] whereas recent research developments in applications of EC to TOD problems can be found in [5].

8.3 Shape Optimization

Shape optimization maintains a fixed topology of structural designs but changes their shape or node locations. Similar to the TOD case, shape optimization problems can be divided into two major groups: continuum SO and discrete SO. Continuum SO addresses shape optimization problems in the context of 2D or 3D *continuum* structures. Traditionally, in continuum SO, "a shape is defined by the oriented boundary curves [2D structures] or boundary surfaces [3D structures] of the body … and the optimal form of these boundaries is computed" [240]. Formal methods for solving continuum SO problems are well-established and extensive literature is available [242-244]. Sensitivity analysis for shape optimization problems is discussed in [245] and application of homogenization method to this problem is offered in [246]. ESO, introduced earlier, has also been used in shape optimization [220]. Evolutionary computation methods have also been applied to solve continuum SO problems. Research on shape optimization of structural members has been conducted by Jenkins [247,248], Richards and Sheppard [249], and Watabe and Okino [250]. Kita and Tanie [251,252] and Annicchiarico and Cerrolaza [253,254] used GAs to optimize the shape of continuum 2D structures through B-spline functions. A GA was used to find optimal locations of knots of B-spline functions. Wibowo and Besari [255] applied GAs to optimize shapes of oval axially symmetric shells. Annicchiarico and Cerrolaza [256] applied GAs to shape optimization of 3D finite element models. Woon et al. [257] investigated alternative encodings of GAs for continuum SO using the actual coordinates of boundary nodes.

Discrete SO methods conduct shape optimization through variations in geometry of discrete truss and frame structures introduced through changes in locations of nodes [258,259]. Various mathematical programming methods have been applied to discrete SO problems, including linear, nonlinear, and dynamic programming [224]. In the case of shape optimization of truss structures, discrete TOD methods using the ground structure have been extended to include optimization of the nodal point locations for a given number and connectivity of nodal points [240]. Initial applications of EC methods to discrete SO problems have been conducted by Grierson and Pak [227,260] in the context of truss structures. Soh and Yang [261] applied fuzzy controlled GAs to optimize the shape of planar and spatial truss structures. Bohnenberger et al. [230] applied GAs to optimize shapes of truss structures in pylons. Keane and Brown [262] used GAs to optimize the shape of a satellite boom with respect to its vibration performance.

SOTA reviews in traditional mathematical approaches to continuum shape optimization problems are presented in [263,264]. Recent developments in formal methods for discrete SO problems can be found in [240,265]. Recent developments in applications of GAs to design of steel structures are described in [2].

8.4 Sizing Optimization

Sizing optimization problems involve finding optimal cross-sections, or dimensions, of elements of a structural system whose topology and shape is fixed. It is the easiest of the three structural optimization problems discussed earlier and relatively well-understood. Research on formal methods of solving these kinds of problems has a long history and extensive literature is available on this topic [266]. First applications of EC to structural optimization problems involved sizing optimization. Lawo and Thierauf [33] used ES to optimize members of a planar six-story frame subjected to earthquake loading. Goldberg and Samtani [102] applied a GA to optimize cross-sections of members of a 10-bar plane truss. Hajela [267,268] investigated cross-section optimization of discrete member trusses using GAs. Deb [269] applied GAs to optimize designs of welded beams. Jenkins [270] proposed a GA-based design environment to optimize plane frame structures. Rajeev and Krishnamoorthy [234] applied GAs to optimize cross-sections of generalized trusses. Recently, Jarmai et al. [271] applied genetic algorithms to design welded I-section frames and compared their performance with other nonlinear optimization algorithms operating in a constrained representation space.

8.5 Historical Perspective

A summary of major applications of EC in structural design since its beginning in the mid 1970's is provided in a chronological order in Table 4. The applications are classified with respect to the application domain, and major EC characteristics, including the representation type, the evolutionary algorithm used, the fitness function, and methods of handling constraints. A chronological classification of the EC applications in structural design clearly shows three major periods in the development of the field:

1. **Period of early explorations (1986-1995)**
   During this initial stage, simple evolutionary algorithms (mainly, if not exclusively ES and GAs, sometimes combined with other traditional optimization methods) were applied to relatively simple structural engineering problems (sizing optimization of simple 2D engineering systems). Researchers focused on using standard design representations, i.e. binary strings and real-valued vectors, single objective fitness functions (usually the minimization of weight) and fairly traditional constraint-handling methods involving various variations of the penalty functions (see section 5.1).
2. **Period of exploration & exploitation (1996-2000)**
   This period can be best characterized as a period of exploring alternative choices for various components of the evolutionary algorithms and improving the process of optimization of more complex design problems. Researchers explored various kinds of representations of engineering systems, including Voronoi-based representations and integer-based representations. Significant research efforts were also focused on tuning the genetic operators to particular problems, e.g. by adapting mutation and crossover rates during the evolutionary design processes. Initial exploration of alternative constraint-handling methods has also been conducted and included e.g. immune networks, behavioral memory, and fuzzy logic. Several multiobjective approaches to structural design problems have been reported as well.
3. **Period of rapid growth (2001-present)**
   Currently, evolutionary computation is a fully recognized structural optimization paradigm and is frequently used not only by researchers but also by practitioners. Nowadays, research efforts are focused on solving much more complex structural design problems and on studying more advanced evolutionary models, including parallel EAs, multiobjective optimization, and variable-length representations, in the context of structural design. Also, initial exploration of the potential of using coevolutionary models is being conducted.

Thus, the field of evolutionary design is far from maturity and there is still a lot to be done. A discussion of the most promising research paths for the future is described in the next section.

9. Discussion and Conclusions

The field of evolutionary design continues to rapidly grow and develop in many exciting new directions. In this section, the authors summarize several of the most promising areas of new research. They can be grouped into the following five classes:

A. Integrated structural design support tools

As the size and complexity of structural problems in the field of evolutionary design continues to increase, there are several scaling-up issues that need to be addressed, including computation time and parallel architectures. Computation time in evolutionary design mostly depends on evaluation of the fitness of generated designs (frequently 90-95%, or more, of computation time). In the past, when computational costs were high, researchers developed a variety of techniques to minimize the computational effort. One of the most popular techniques involved separation of the stages of conceptual, preliminary, and detailed design, and developing separate tools for each stage [6]. Nowadays, however, the cost of computation continues to decrease and this trend is likely to persist in the future. Also, parallel computer architectures are now readily available. Considering the fact that EAs have a natural mapping onto parallel architectures, it is the authors' belief that computational costs should not be the primary factor in developing new integrated evolutionary-based structural design support tools. These tools will treat all the stages of a design process as phases of a single integrated design process. Research efforts in this direction are led by Parmee and co-workers [42].

B. Open-ended representations

A appropriate representation of an engineering system is one of the key issues in any structural design application. Today, it becomes even more important because the increased complexity of considered design problems raises some difficult internal EA issues on how to best represent and evolve complex designs [6]. Another motivation comes from the fact that there is an emerging trend to apply evolutionary design techniques not only to strictly optimization tasks but rather this technique is being gradually more and more useful in finding novel design concepts. Both issues lead to open-ended representations (see section 4) which don't encode entire designs but rather rules on how to construct these designs [272,273]. Representations of this type are also inspired by the processes occurring in nature, where we observe evolution manipulating the genetic plans for complex objects rather than the objects themselves. The organisms are then built from the plans via a developmental process called morphogenesis.

C. Alternative constraint-handling methods

Almost every structural design problem involves some kind of constraints. Up to very recently, various variations of penalty functions were virtually the only method of handling constraints. On the other hand, a number of applications showed that there are many difficulties associated with this approach when applied to highly constrained optimization problems. Studies focused on estimating a true potential of alternative constraint-handling methods (discussed in section 5) constitute another promising area for future research of vital importance to structural design.

D. Multiobjective structural design

Structural design problems are inherently multiobjective and often involve a relatively large number of conflicting criteria. So far, research in evolutionary structural design concentrated almost exclusively, with few notable exceptions, on single objective problems. At the same time, the field of evolutionary multiobjective optimization provides new and efficient methods, described in section 6, of solving these types of problems. Multiobjective structural design may become one of the most promising areas of research in structural design, particularly when not a single optimal design solution is sought but rather a set of alternative optimal designs.

E. Coevolutionary structural design

Coevolutionary design is an emerging area of research with many unanswered questions. There is a lot to be done to understand the true potential of this paradigm in structural design. Initial findings coming from evolutionary computation community suggest that coevolutionary models might be particularly suitable for complex design spaces that can be relatively well decomposed and when the major goal is not the optimality of design solutions in a global sense, but rather their robustness (design for reliability) [188]. As stated earlier, very little has been done in this area and it is potentially one of the most promising paths of future research.

Evolutionary computation is becoming a computational paradigm that is increasingly attractive for civil engineers. This phenomenon is a part of the ongoing Information Technology Revolution and is directly related to the changing nature of the use of computers from the exclusive applications in the analysis to their holistic use in design, including conceptual design, integrated design, etc. Research on evolutionary computation in structural design is now fully recognized research activity and reported in the leading journals in this area, including:

- Journal of Computing in Civil Engineering
- Journal of Structural Engineering
- Computers & Structures
- Advances in Engineering Software
- Structural and Multidisciplinary Optimization
- Engineering Optimization
- AI in Engineering Design

Also, research findings considering evolutionary computation in structural design are regularly presented during several respected international conferences and workshops:

- International Conference on Adaptive Computing in Design and Manufacture (ACDM)
- World Congress of Structural and Multidisciplinary Optimization (WCSMO)
- ASME Conference On Design
- AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference (SDM)
- Genetic and Evolutionary Computation Conference (GECCO)
- International Workshop on Information Technology in Civil Engineering (ASCE)
- International Workshop of the European Group for Intelligent Computing in Engineering (EG-ICE)

After years of research, the field of evolutionary computation in structural design has generated significant amount of both theoretical and empirical knowledge, which justifies the transfer of knowledge, technologies and tools from the academia to the practicing structural designers. Finally, there is a chance to close the existing gap between the needs of designers and the evolutionary design scholars.

Evolutionary design is a fascinating interdisciplinary research area, where concepts from computer science, heuristics, design and inventive engineering and structural engineering are integrated and transformed in the process. In this way, a new understanding of structural design emerges in the tradition of synesthesia proposed by Leonardo DaVinci and still continued by creative scholars and practitioners.

10. References

[1] De Jong, K. A. (to appear). *Evolutionary computation: a unified approach*. Cambridge, MA: MIT Press.

[2] Pezeshk, S. (2002). State of the art on the use of genetic algorithms in design of steel structures. In S. Burns (Ed.), *Recent Advances in Optimal Structural Design*. Reston, VA: American Society of Civil Engineers

[3] Grierson, D. E., & Khajehpour, S. (2002). Conceptual design optimization of engineering structures. In S. Burns (Ed.), *Recent advances in optimal structural design*. Reston, VA: American Society of Civil Engineers, 81-95.

[4] Cheng, F. Y. (2002). Multiobjective optimum design of seismic-resistant structures. In S. Burns (Ed.), *Recent advances in optimal structural design*. Reston, VA: American Society of Civil Engineers, 241-255.

[5] Hajela, P., & Vittal, S. (2000). *Evolutionary computing and topology optimization: a state of the art assessment.* In Proceedings of the NATO Advanced Research Workshop on Topology Optimization, Budapest, Hungary.

[6] Arciszewski, T., & De Jong, K. A. (2001). *Evolutionary computation in civil engineering: research frontiers.* In B. H. V. Topping (Ed.), Proceedings of the Eight International Conference on Civil and Structural Engineering Computing, Eisenstadt, Vienna, Austria.

[7] Shaw, D., Miles, J. C., & Gray, A. (2003). *Genetic programming within civil engineering: a review.* In O. Ciftcioglu & E. Dado (Eds.), Proceedings of the 10th International Workshop of the European Group for Intelligent Computing in Engineering (EG-ICE), Delft, The Netherlands, 29-39.

[8] Darwin, C. (1859). *The origin of species by means of natural selection*. London: J. Murray.

[9] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, Michigan: University of Michigan Press.

[10] Luke, S. (2000). *Issues in scaling genetic programming: breeding strategies, tree generation, and code bloat.* Ph.D. Dissertation, Department of Computer Science, University of Maryland, College Park, Maryland.
[11] Rechenberg, I. (1965). *Cybernetic solution path of an experimental problem* (Vol. Library Translation 1122). Farnborough, UK: Royal Aircraft Establishment.
[12] Schwefel, H.-P. (1965). *Kybernetische Evolution als Strategie der experimentelen Forschung in der Stromungstechnik.* Master's thesis, Hermann Föttinger Institute for Hydrodynamics, Technical University of Berlin.
[13] Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial Intelligence through simulated evolution*. Chichester, UK: John Wiley.
[14] Koza, J. R. (1992). *Genetic programming : on the programming of computers by means of natural selection*. Cambridge, Mass.: MIT Press.
[15] Eshelman, L. J. (1991). *The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination.* In G. J. E. Rawlins (Ed.), Proceedings of the Second Workshop on Foundations of Genetic Algorithms, Vail, CO, USA, 265--283.
[16] Dasgupta, D., & MacGregor, D. (1991). *A structured genetic algorithm* (No. IKBS-2-91): University of Strathclyde, UK.
[17] Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithms. Continuous parameter optimization. *Evolutionary Computation, 1*(1), 25 - 49.
[18] Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems, 3*(5), 493-530.
[19] Whitley, L. D. (1989). *The GENITOR algorithm and selection pressure: why ranked-based allocation of reproductive trials is best.* In J. D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89), Fairfax, VA, USA, 239–255.
[20] Grefenstette, J. J., & Baker, J. E. (1989). *How genetic algorithms work: a critical look at implicit parallelism.* In J. D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89), Fairfax, VA, USA, 20-27.
[21] Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-modellen mittels der Evolutionsstrategie*. Basel: Birkhaeuser Verlag.
[22] Spears, W. M. (2000). *Evolutionary algorithms: the role of mutation and recombination*. Berlin ; New York: Springer.
[23] Fogarty, T. C. (1989). *Varying the probability of mutation in genetic algorithm.* In J. D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89), Fairfax, VA, USA, 104-109.
[24] Fairley, A. (1991). *Comparison of methods of choosing the crossover point in the genetic crossover operation* (Technical Report). Liverpool, UK: University of Liverpool.
[25] Schaffer, J. D., & Eshelman, L. J. (1991). *On crossover as an evolutionarily viable strategy.* In R. K. Belew & L. B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91), San Diego, CA, USA, 61-68.
[26] Cohoon, J. P., Hegde, S. U., Martin, W. N., & Richards, D. S. (1987). *Punctuated equilibria: a parallel genetic algorithm.* In J. J. Grefenstette (Ed.), Proceedings of the Second International Conference on Genetic Algorithms (ICGA'87), Cambridge, MA, USA, 148–154.
[27] Potter, M. A., & De Jong, K. A. (2000). Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolutionary Computation, 8*(1), 1-29.
[28] Angeline, P. J., & Pollack, J. B. (1993). *Competitive environments evolve better solutions for complex tasks.* In S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, USA, 264–270.
[29] Bentley, P. J. (1999). An introduction to evolutionary design by computers. In P. J. Bentley (Ed.), *Evolutionary Design by Computers*. San Francisco, CA: Morgan Kaufmann Publishers
[30] Parmee, I. C. (1999). Exploring the design potential of evolutionary search, exploration and optimisation. In P. J. Bentley (Ed.), *Evolutionary Design by Computers*. London: Academic Press Ltd.
[31] Rechenberg, I. (1973). *Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart-Bad Cannstatt: Frommann-Holzboog.
[32] Hoeffler, A., Leysner, U., & Weidermann, J. (1973). *Optimization of the layout of trusses combining strategies based on Mitchel's theorem and on biological principles of evolution.* In Proceedings of the 2nd Symposium on Structural Optimization, Milan, Italy.

[33] Lawo, M., & Thierauf, G. (1982). Optimal design for dynamic stochastic loading: a solution by random search. In *Optimization in Structural Design*. University of Siegen: Bibl. Inst. Mannheim, 346-352.

[34] Goldberg, D. E. (1987). Computer-aided gas pipeline operation using genetic algorithms and rule learning, part I: Genetic algorithm in pipeline optimization. *Engineering with Computers*, 47-58.

[35] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass.: Addison-Wesley Pub. Co.

[36] Bentley, P. J. (Ed.). (1999). *Evolutionary design by computers*. San Francisco, CA: Morgan Kaufmann Publishers.

[37] Bentley, P. J., & Corne, D. W. (Eds.). (2002). *Creative evolutionary systems*. San Francisco, CA: Morgan Kaufmann Publishers.

[38] Dasgupta, D., & Michalewicz, Z. (Eds.). (1997). *Evolutionary algorithms in engineering applications*. Berlin, Heidelberg: Springer-Verlag.

[39] Cvetkovic, D., & Parmee, I. C. (1999). *Genetic algorithms based systems for conceptual engineering design.* In U. Lindemann, H. Birkhofer, H. Meerkamm & S. Vajna (Eds.), Proceedings of the 12th International Conference on Engineering Design ICED'99, München, Germany, 1035-1038.

[40] Coello Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems*. New York: Kluwer Academic.

[41] Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization*. New York: Wiley.

[42] Parmee, I. C. (2001). *Evolutionary and adaptive computing in engineering design*. London, New York: Springer.

[43] Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: Wiley.

[44] Parmee, I. C. (Ed.). (2002). *Adaptive computing in design and manufacture V*. London, New York: Springer-Verlag.

[45] Chawdhry, P., Roy, R., & Pant, R. (Eds.). (1998). *Soft computing in engineering design and manufacturing*. London ; New York: Springer.

[46] Gero, J. S. (1996). Computers and creative design. In M. Tan & R. Teh (Eds.), *The Global Design Studio*: National University of Singapore, 11-19.

[47] Boden, M. A. (1992). *The creative mind: myths and mechanisms*. New York: Basic Books.

[48] Arciszewski, T., & Michalski, R. S. (1984). *Inferential design theory.* In J. S. Gero & F. Sudweeks (Eds.), Proceedings of the Third International Conference on Artificial Intelligence in Design, Lausanne, Switzerland, 295-309.

[49] Arciszewski, T., Michalski, R. S., & Wnek, J. (1995). Constructive induction: the key to design creativity. In J. S. Gero & M. L. Maher (Eds.), *Preprints of the Third International Round-Table Conference on Computational Models of Creative Design*. Heron Island, Queensland, Australia, 397-426.

[50] Rosenman, M. (1997). The generation of form using evolutionary approach. In D. Dasgupta & Z. Michalewicz (Eds.), *Evolutionary algorithms in engineering applications*. Berlin New York: Springer, 69-86.

[51] Altshuller, G. (1969). *Algorithm of invention*. Moscow: Moskowskij Raboczij Publishing House.

[52] Altshuller, G. (1999). *The innovation algorithm: TRIZ, systematic innovation and technical creativity*: Technical Innovation Center.

[53] Rosenman, M., & Gero, J. S. (1999). Evolving designs by generating useful complex gene structures. In P. J. Bentley (Ed.), *Evolutionary design by computers*. San Francisco, CA: Morgan Kaufmann Publishers

[54] Bentley, P. J. (2000). *Exploring component-based representations.* In I. C. Parmee (Ed.), Proceedings of the Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM'2000), University of Plymouth, UK, 161-172.

[55] Hornby, G. S. (2003). *Generative representations for evolutionary design automation.* Ph.D. Dissertation, Department of Computer Science, Brandeis University, Waltham, MA, USA.

[56] Bentley, P. J., & Kumar, S. (1999). *Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem.* In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela & R. E. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), Orlando, Florida, USA, 35-43.

[57] Wolfram, S. (2002). *A new kind of science*. Champaign, IL: Wolfram Media.

[58] Gero, J. S. (1992). Creativity, emergence and evolution in design. *Preprints Computational Models of Creative Design*, 1-28.

[59] Kicinger, R., De Jong, K. A., & Arciszewski, T. (2002). Long term versus short term evolutionary design. In M. Schnellenbach-Held & H. Denk (Eds.), *Advances in Intelligent Computing in Engineering. Proceedings of the 9th International Workshop of the European Group for Intelligent Computing in Engineering*. Darmstadt, Germany: VDI Verlag, 184-195.

[60] Gero, J. S., & Schnier, T. (1995). Evolving representations of design cases and their use in creative design. In J. S. Gero, M. L. Maher & F. Sudweeks (Eds.), *Preprints Computational Models of Creative Design*. Syndey, Australia: Key Center of Design Computing, University of Sydney, 343-368.

[61] Arciszewski, T., De Jong, K. A., & Vyas, H. (1999). *Inventive design in structural engineering: evolutionary computation approach*. In B. Kumar & B. H. V. Topping (Eds.), Proceedings of the Fifth International Conference on the Applications of AI to Civil and Structural Engineering, Oxford, England, 1-9.

[62] Bentley, P. J. (1999). From coffee tables to hospitals: generic evolutionary design. In P. J. Bentley (Ed.), *Evolutionary design by computers*. San Francisco, CA: Morgan Kaufmann Publishers

[63] Parmee, I. C. (1995). *Diverse evolutionary search for preliminary whole system design.* In Proceedings of the 4th International Conference on AI in Civil and Structural Engineering, Cambridge University.

[64] Pahl, G., & Beitz, W. (1996). *Engineering design: a systematic approach*. New York: Springer Verlag.

[65] Parmee, I. C. (1996). *The maintenance of search diversity for effective design space decomposition using cluster-oriented genetic algorithms (COGAs) and multi-agent strategies (GAANT).* In Proceedings of the Adaptive Computing in Engineering Design and Control, University of Plymouth, UK.

[66] Bilchev, G., & Parmee, I. C. (1995). *The ant colony metaphor for searching continuous design spaces.* In T. C. Fogarty (Ed.), Proceedings of the Evolutionary Computing, Sheffield, UK, 25-39.

[67] Colorni, A., Dorigo, M., & Maniezzo, V. (1992). *An investigation of some properties of the ant algorithm.* In R. Männer & B. Manderick (Eds.), Proceedings of the Second International Conference on Parallel Problem Solving from Nature (PPSN-II), Brussels, Belgium, 515-526.

[68] Michalewicz, Z., Dasgupta, D., Le Riche, R. G., & Schoenauer, M. (1996). Evolutionary algorithms for constrained engineering problems. *Computers and Industrial Engineering Journal, 30*(4), 851-830.

[69] Parmee, I. C. (Ed.). (1998). *Adaptive computing in design and manufacture : the integration of evolutionary and adaptive computing technologies with product/system design and realisation.* London; New York: Springer-Verlag.

[70] Parmee, I. C. (1998). Genetic algorithms, and hydropower system design. *Computer-Aided Civil and Infrastructure Engineering, 13*(1), 31-41.

[71] Vekeria, H. D., & Parmee, I. C. (1996). *The use of a co-operative multi-level CHC GA for structual shape optimisation.* In Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany.

[72] Bonham, C. R., & Parmee, I. C. (1999). *Improving the performance of cluster oriented genetic algorithms (COGAs).* In Proceedings of the Congress on Evolutionary Computation (CEC'1999), Washington, DC, USA, 554-561.

[73] Dym, C. L. (1994). *Engineering design: a synthesis of views*. New York, NY: Cambridge University Press.

[74] Baron, P., Fisher, R., Mill, F., Sherlock, A., & Tuson, A. (1997). *A voxel-based representation for the evolutionary shape optimization of a simplified beam: a case-study of a problem-centered approach to genetic operator design.* In Proceedings of the 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2).

[75] Kane, C., & Schoenauer, M. (1995). Genetic operators for two-dimensional shape optimization. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer & D. Snyers (Eds.), *Artificial Evolution* (Vol. Lecture Notes in Computer Science 1063): Springer Verlag

[76] Kane, C., & Schoenauer, M. (1996). Topological optimum design using genetic algorithms. *Control and Cybernetics, 25*(5), 1059-1088.

[77] Roston, G. P. (1994). *A genetic methodology for configuration design.* Ph.D. Dissertation, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA.

[78] Funes, P., & Pollack, J. B. (1999). Computer evolution of buildable objects. In P. J. Bentley (Ed.), *Evolutionary design by computers*. San Francisco, CA: Morgan Kaufmann Publishers

[79] Bentley, P. J. (1996). *Generic evolutionary design of solid objects using a genetic algorithm.* Ph.D. Dissertation, Division of Computing and Control Systems, Department of Engineering, University of Huddersfield, Queensgate, Huddersfield, UK.

[80] Shea, K., Cagan, J., & Fenves, S. J. (1997). A shape annealing approach to optimal truss design with dynamic grouping of members. *Journal of Mechanical Design*, *119*, 388-394.

[81] Schmidt, L. C., & Cagan, J. (1998). Optimal configuration design: an integrated approach using grammars. *Journal of Mechanical Design*, *120*(1), 2-9.

[82] Grabska, E. (1993). *Graphs and designing.* In H. J. Schneider & H. Ehrig (Eds.), Proceedings of the International Workshop on Graph Transformations in Computer Science, Dagstuhl Castle, Germany, 188-202.

[83] Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, *7*(3), 343-351.

[84] Shai, O. (2001). Combinatorial representations in structural analysis. *Journal of Computing in Civil Engineering*, *15*(3), 193-207.

[85] Frazer, J. (1995). *An evolutionary architecture*. London: Architectural Association Publications.

[86] Hajela, P., & Kim, B. (1999). *GA based learning in cellular automata models for structural analysis.* In Proceedings of the 3rd World Congress on Structural and Multidisciplinary Optimization, Niagara Falls, NY.

[87] Coates, P. (1997). *Using genetic programming and L-systems to explore 3D design worlds.* In R. Junge (Ed.), Proceedings of the CAAD Futures '97, Munich, Germany.

[88] Jacob, C. (1994). *Genetic L-system programming.* In Y. Davidor, H.-P. Schwefel & R. Männer (Eds.), Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN-III), Jerusalem, Israel, 334-343.

[89] Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms: I. Representation. *Computers and Industrial Engineering, 30*(4), 983-997.

[90] Sendhoff, B., Kreutz, M., & Seelen, W. v. (1997). *A condition for the genotype-phenotype mapping: causality*. In T. Bäck (Ed.), Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97), East Lansing, MI, USA, 354-361.

[91] Popovici, E. (2003). *The bleeding edge of inventive design* (Computer Science Technical Report). Fairfax, VA: George Mason University.

[92] De Jong, E. D., & Oates, T. (2002). *A coevolutionary approach to representation development.* In E. D. De Jong & T. Oates (Eds.), Proceedings of the ICML-2002 Workshop on Development of Representations, The University of New South Wales, Sydney, Australia.

[93] Dorn, W. C., Gomory, R. E., & Greenberg, H. J. (1964). Automatic design of optimal structures. *Journal de Mecanique*, *3*, 25-52.

[94] Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd rev. and extended ed.). Berlin ; New York: Springer-Verlag.

[95] Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford, New York: Oxford University Press.

[96] Rothlauf, F. (2002). *Representations for genetic and evolutionary algorithms*. Heidelberg New York: Physica-Verlag.

[97] Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, *191*, 1245-1287.

[98] Michalewicz, Z. (1995). *A survey of constraint handling techniques in evolutionary computation methods.* In Proceedings of the 4th Annual Conference on Evolutionary Programming, Cambridge, MA, 135-155.

[99] Courant, R. (1943). Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of American Mathematical Society*, *49*, 1-23.

[100] Caroll, C. W. (1961). The created response surface technique for optimizing nonlinear restrained systems. *Operations Research*, *9*, 169-184.

[101] Fiacco, A. V., & McCormick, G. P. (1968). Extensions to SUMT for nonlinear programming: equality constraints and extrapolation. *Management Science*, *12*(11), 816-828.

[102] Goldberg, D. E., & Samtani, M. (1986). *Engineering optimization via genetic algorithm.* In Proceedings of the Ninth Conference on Electronic Computation, University of Alabama, Birmingham, 471-482.

[103] Richardson, J. T., Palmer, M. R., Liepins, G. E., & Hilliard, M. R. (1989). *Some guidelines for genetic algorithms with penalty functions.* In J. D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89), Fairfax, VA, USA, 191-197.

[104] Carlson, S. E. (1995). *A general method for handling constraints in genetic algorithms.* In Proceedings of the Second Annual Joint Conference on Information Science, 663-667.

[105] Joines, J. A., & Houck, C. R. (1994). *On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's.* In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel & H. Kitano (Eds.), Proceedings of the First IEEE International Conference on Evolutionary Computation (ICEC'94), Orlando, FL, USA, 579-584.

[106] Michalewicz, Z., & Attia, N. F. (1994). *Evolutionary optimization of constrained problems.* In A. V. Sebald & L. J. Fogel (Eds.), Proceedings of the Third Annual Conference on Evolutionary Programming, San Diego, CA, USA, 98-108.

[107] Bean, J. C., & Hadj-Alouane, A. B. (1992). *A dual genetic algorithm for bounded integer programs* (Technical Report No. TR 92-53): Department of Industrial and Operations Engineering, University of Michingan.

[108] Hadj-Alouane, A. B., & Bean, J. C. (1997). A genetic algorithm for the multiple-choice integer program. *Operations Research, 45*, 92-101.

[109] Smith, A. E., & Tate, D. M. (1993). *Genetic optimization using a penalty function.* In S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, USA, 499-503.

[110] Rasheed, K. (1998). *An adaptive penalty approach for constrained genetic-algorithm optimization.* In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba & R. L. Riolo (Eds.), Proceedings of the Third Annual Genetic Programming Conference, Madison, Wisconsin, USA, 584-590.

[111] Nanakorn, P., & Meesomklin, K. (2001). An adaptive penalty function in genetic algorithms for structural design optimization. *Computers & Structures, 79*(29-30), 2527-2539.

[112] Coello Coello, C. A. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*(2), 113-127.

[113] Schwefel, H.-P. (1981). *Numerical optimization of computer models.* Chichester, UK: John Wiley & Sons.

[114] Bean, J. C. (1994). Genetics and random keys for sequencing and optimization. *ORSA Journal on Computing, 6*, 154-160.

[115] Davidor, Y. (1989). *Analogous crossover.* In J. D. Schaffer (Ed.), Proceedings of the Third International Conference On Genetic Algorithms (ICGA'89), Fairfax, VA, USA, 98-103.

[116] Kowalczyk, R. (1997). *Constraint consistent genetic algorithms.* In Proceedings of the Fourth IEEE International Conference on Evolutionary Computation (ICEC'97), Indianapolis, USA, 343 -348.

[117] Schoenauer, M., & Michalewicz, Z. (1996). *Evolutionary computation at the edge of feasibility.* In H.-M. Voigt, W. Ebeling, I. Rechenberg & H.-P. Schwefel (Eds.), Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN-IV), Berlin, Germany, 245-254.

[118] Schoenauer, M., & Michalewicz, Z. (1997). *Boundary operators for constrained parameter optimization problems.* In T. Bäck (Ed.), Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97), East Lansing, MI, USA, 320-329.

[119] Michalewicz, Z. (2000). Decoders. In T. Bäck, D. B. Fogel & Z. Michalewicz (Eds.), *Evolutionary computation 2: advanced algorithms and operators* (Vol. 2). Bristol and Philadelphia: Institute of Physics Publishing, 49-55.

[120] Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation, 7*(1), 19-44.

[121] Kim, D. G. (1998). *Riemann mapping based constraint handling for evolutionary search.* In Proceedings of the 1998 ACM Symposium on Applied Computing, Atlanta, GA, USA, 379-385.

[122] Michalewicz, Z. (2000). Repair algorithms. In T. Bäck, D. B. Fogel & Z. Michalewicz (Eds.), *Evolutionary computation 2: advanced algorithms and operators* (Vol. 2). Bristol and Philadelphia: Institute of Physics Publishing, 56-61.

[123] Liepins, G. E., & Vose, M. D. (1990). Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence, 2*, 101-115.
[124] Mühlenbein, H. (1992). Parallel genetic algorithms in combinatorial optimization: new developments in their interfaces. In O. Balci, R. Sharda & S. A. Zenios (Eds.), *Computer Science and Operations Research*. New York: Pergamon Press, 441-456.
[125] Tate, D. M., & Smith, A. E. (1995). A genetic approach to the quadratic assignment problem. *Computers and Operations Research, 22*(1), 73-78.
[126] Michalewicz, Z., & Nazhiyath, G. (1995). *Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints.* In D. B. Fogel (Ed.), Proceedings of the Second IEEE International Conference on Evolutionary Computation (ICEC'95), Perth, Australia, 647-651.
[127] Liepins, G. E., & Potter, W. D. (1991). A genetic algorithm approach to multiple-fault diagnosis. In L. Davis (Ed.), *Handbook of genetic Algorithms*. New York: Van Nostrand Reinhold, 237-250.
[128] Nakano, R., & Yamada, T. (1991). *Conventional genetic algorithm for job shop problems.* In R. K. Belew & L. B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91), San Diego, CA, USA, 474-479.
[129] Orvosh, D., & Davis, L. (1994). *Using a genetic algorithm to optimize problems with feasibility constraints.* In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel & H. Kitano (Eds.), Proceedings of the First IEEE International Conference on Evolutionary Computation (ICEC'94), Orlando, FL, USA, 548 -553.
[130] Kicinger, R., Arciszewski, T., & De Jong, K. A. (2003). Evolutionary designing of steel structures in tall buildings. *Journal of Computing in Civil Engineering*(tentatively approved).
[131] Paredis, J. (1994). *Co-evolutionary constraints satisfaction.* In Y. Davidor, H.-P. Schwefel & R. Männer (Eds.), Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN-III), Jerusalem, Israel, 46-55.
[132] Powell, D. J., & Skolnick, M. M. (1993). *Using genetic algorithms in engineering design optimization with non-linear constraints.* In S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, USA, 424-431.
[133] Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186*(2-4), 311-338.
[134] Schoenauer, M., & Xanthakis, S. (1993). *Constrained GA optimization.* In S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, USA, 573-580.
[135] Surry, P. D., Radcliffe, N. J., & Boyd, I. D. (1995). *A multi-objective approach to constrained optimization of gas supply networks: the COMOGA method.* In T. C. Fogarty (Ed.), Proceedings of the AISB-95 Workshop on Evolutionary Computing, Sheffield, UK, 166-180.
[136] Surry, P. D., & Radcliffe, N. J. (1997). The COMOGA method: constrained optimisation by multi-objective genetic algorithms. *Control and Cybernetics, 26*(3).
[137] Parmee, I. C., & Purchase, G. (1994). *The development of a directed genetic search technique for heavily constrained design spaces.* In I. C. Parmee (Ed.), Proceedings of the First International Conference on Adaptive Computing in Engineering Design and Control, Plymouth, UK, 97-102.
[138] Coello Coello, C. A. (2000). Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization, 32*(3), 275-308.
[139] Coello Coello, C. A. (2000). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems, 17*, 319-346.
[140] Adeli, H., & Cheng, N. T. (1994). Augmented Lagrangian genetic algorithm for structural optimization. *Journal of Structural Engineering, 7*(3), 104-118.
[141] Myung, H., Kim, J.-H., & Fogel, D. B. (1995). *Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems.* In J. R. McDonnell, R. G. Reynolds & D. B. Fogel (Eds.), Proceedings of the Fourth Annual Conference on Evolutionary Programming, Cambridge, MA, USA, 449-463.
[142] Kim, J.-H., & Myung, H. (1997). Evolutionary programming techniques for constrained optimization problems. *IEEE Transactions on Evolutionary Computation, 1*(2), 129 -140.

[143] Le, T. V. (1995). *A fuzzy evolutionary approach to constrained optimization problems.* In Proceedings of the Second IEEE International Conference on Evolutionary Computation (ICEC'95), Perth, Australia, 274-278.

[144] Le, T. V. (1996). *A fuzzy evolutionary approach to constrained optimisation problems.* In T. Fukuda & T. Furuhashi (Eds.), Proceedings of the Third IEEE International Conference on Evolutionary Computation (ICEC'96), Nagoya, Japan, 274-278.

[145] Forrest, S., & Perelson, A. S. (1990). *Genetic algorithms and the immune system.* In H.-P. Schwefel & R. Männer (Eds.), Proceedings of the First International Conference on Parallel Problem Solving from Nature (PPSN-I), Dortmund, Germany, 320-325.

[146] Smith, R. E., Forrest, S., & Perelson, A. S. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation, 1*(2), 127-149.

[147] Hajela, P., & Lee, J. (1996). Constrained genetic search via schema adaptation. An immune network solution. *Structural Optimization, 12*(1), 11-15.

[148] Hajela, P., & Lee, J. (1995). *Constrained genetic search via schema adaptation. An immune network solution.* In N. Olhoff & G. I. N. Rozvany (Eds.), Proceedings of the First World Congress of Structural and Multidisciplinary Optimization (WCSMO-1), Goslar, Germany, 915-920.

[149] Yoo, J., & Hajela, P. (1999). Immune network simulations in multicriterion design. *Structural and Multidisciplinary Optimization, 18*(2-3), 85-94.

[150] Reynolds, R. G. (1994). *An introduction to cultural algorithms.* In A. V. Sebald & L. J. Fogel (Eds.), Proceedings of the Third Annual Conference on Evolutionary Programming, Singapore, 131-139.

[151] Reynolds, R. G., Michalewicz, Z., & Cavaretta, M. J. (1995). *Using cultural algorithms for constraint handling in Genocop.* In J. R. McDonnell, R. G. Reynolds & D. B. Fogel (Eds.), Proceedings of the Fourth Annual Conference on Evolutionary Programming, Cambridge, MA, USA, 289-305.

[152] Chung, C.-J., & Reynolds, R. G. (1996). *A testbed for solving optimization problems using cultural algorithms.* In L. J. Fogel, P. J. Angeline & T. Bäck (Eds.), Proceedings of the Fifth Annual Conference on Evolutionary Programming, San Diego, CA, USA.

[153] Colorni, A., Dorigo, M., & Maniezzo, V. (1991). *Distributed optimization by ant colonies.* In Proceedings of the First European Conference on Artificial Life (ECAL'91), Paris, France.

[154] Bilchev, G., & Parmee, I. C. (1996). *Constrained and multi-modal optimisation with an ant colony search model.* In I. C. Parmee & M. J. Denham (Eds.), Proceedings of the Second International Conference on Adaptive Computing in Engineering Design and Control, Plymouth, UK.

[155] Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation, 4*(1), 1-32.

[156] Coello Coello, C. A. (1999). A comprehensive survey of evolutionary-based multi-objective optimization techniques. *Knowledge and Information Systems, 1*(3), 269-308.

[157] Pareto, V. (1896). *Cours D'Economie Politique* (Vol. I and II). Lausanne: F. Rouge.

[158] Deb, K. (1999). Evolutionary algorithms for multi-criterion optimization in engineering design. In K. Miettinen, M. M. Makela, P. Neittaanmaki & J. Periaux (Eds.), *Evolutionary algorithms in engineering and computer science : recent advances in genetic algorithms, evolution strategies, evolutionary programming, genetic programming, and industrial applications*. Chichester; New York: John Wiley & Sons

[159] Chankong, V., & Haimes, Y. Y. (1983). *Multiobjective decision making: theory and methodology*. New York: North Holland.

[160] Haimes, Y. Y., Lasdon, L. S., & Wismer, D. A. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics, 1*(3), 296-297.

[161] Coello Coello, C. A. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys, 32*(2), 109-143.

[162] Rosenberg, R. S. (1967). *Simulation of genetic populations with biochemical properties.* Ph.D. DissertationUniversity of Michigan, Ann Harbor, Michigan.

[163] Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms.* Ph.D. DissertationVanderbilt University, Nashville, TN.

[164] Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, New York: John Wiley & Sons.

[165] Syswerda, G., & Palmucci, J. (1991). *The application of genetic algorithms to resource scheduling.* In R. K. Belew & L. B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91), San Diego, CA, USA, 502-508.

[166] Yang, X., & Gen, M. (1994). *Evolution program for bicriteria transportation problem.* In M. Gen & T. Kobayashi (Eds.), Proceedings of the 16th International Conference on Computers and Industrial Engineering, Ashikaga, Japan, 451-454.

[167] Jakob, W., Gorges-Schleuter, M., & Blume, C. (1992). *Application of genetic algorithms to task planning and learning.* In R. Männer & B. Manderick (Eds.), Proceedings of the Second International Conference on Parallel Problem Solving from Nature (PPSN-II), Brussels, Belgium, 293-302.

[168] Schaffer, J. D. (1985). *Multiple objective optimization with vector evaluated genetic algorithms.* In J. J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms (ICGA'85), Pittsburgh, PA, USA, 93-100.

[169] Ritzel, B. J., Eheart, J. W., & Ranjithan, S. (1994). Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research, 30*(5), 1589-1603.

[170] Cvetkovic, D., Parmee, I. C., & Webb, E. (1998). Multi-objective optimisation and preliminary airframe design. In I. C. Parmee (Ed.), *The Integration of Evolutionary and Adaptive Computing Technologies with Product/System Design and Realisation*. Plymouth, UK: Springer-Verlag, 255-267.

[171] Charnes, A., & Cooper, W. W. (1961). *Management models and industrial applications of linear programming*. New York: John Wiley.

[172] Chen, Y. L., & Liu, C. C. (1994). Multiobjective VAR planning using the goal-attainment method. *IEE Proceedings on Generation, Transmission and Distribution, 141*(3), 227-232.

[173] Hajela, P., & Lin, C.-Y. (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization*(4), 99-107.

[174] Coello Coello, C. A., & Christiansen, A. D. (1998). Two new GA-based methods for multiobjective optimization. *Civil Engineering Systems, 15*(3), 207-243.

[175] Coello Coello, C. A., & Christiansen, A. D. (2000). Multiobjective optimization of trusses using genetic algorithms. *Computers & Structures, 75*(6), 647-660.

[176] Fonseca, C. M., & Fleming, P. J. (1993). *Genetic algorithms for multi-objective optimization: formulation, discussion, and generalization.* In S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, USA, 416-423.

[177] Chipperfield, A. J., & Fleming, P. J. (1995). *Gas turbine engine controller design using multiobjective genetic algorithms.* In A. M. S. Zalzala (Ed.), Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), Halifax Hall, University of Sheffield, UK.

[178] Obayashi, S. (1998). Pareto genetic algorithm for aerodynamic design using the Navier-Stokes equations. In D. Quagliarella, J. Periaux, C. Poloni & G. Winter (Eds.), *Genetic algorithms and evolution strategies in engineering and computer science: recent advances and industrial applications*. Chichester, England: John Wiley & Sons, 245-266.

[179] Obayashi, S. (2002). Pareto solutions of multipoint design of supersonic wings using evolutionary algorithms. In I. C. Parmee (Ed.), *Adaptive Computing in Design and Manufacture V*. London: Springer-Verlag, 3-16.

[180] Grierson, D. E., & Khajehpour, S. (2002). Method for conceptual design applied to office buildings. *Journal of Computing in Civil Engineering, 16*(2), 83-103.

[181] Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation, 2*(3), 221-248.

[182] Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). *Fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II.* In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo Guervós & H.-P. Schwefel (Eds.), Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN-VI), Paris, France, 849-858.

[183] Hamda, H., Roudenko, O., & Schoenauer, M. (2002). *Multi-objective evolutionary topological optimum design.* In I. C. Parmee (Ed.), Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002), University of Exeter, Devon, UK, 121-132.

[184] Deb, K., & Goel, T. (2001). *A hybrid multi-objective evolutionary approach to engineering shape design.* In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello & D. W. Corne (Eds.), Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO'2001), Zurich, Switzerland, 385-399.

[185] Horn, J., & Nafpliotis, N. (1993). *Multiobjective optimization using the niched Pareto genetic algorithm* (Technical Report No. IlliGAl Report 93005). Urbana, Illinois, USA: University of Illinois at Urbana-Champaign.

[186] Zitzler, E., & Thiele, L. (1998). *An evolutionary algorithm for multi-objective optimization: the strength Pareto approach* (No. Technical Report 43). Zurich, Switzerland: Computer Engineering and Communication Networks Laboratory, Swiss Federal Institute of Technology.

[187] Van Veldhuizen, D. A., & Lamont, G. B. (1998). *Multi-onjective evolutionary algorithm research: a history and analysis* (No. TR-98-03). Wright-Patterson AFB, Ohio: Department of Electrical and Computer Engineering, Air Force Institute of Technology.

[188] Wiegand, R. P. (2003). *An analysis of cooperative coevolutionary algorithms.* Ph.D. Dissertation, Department of Computer Science, George Mason University, Fairfax, VA, USA.

[189] Rosin, C. D., & Belew, R. K. (1996). *New methods for competitive coevolution* (Technical Report No. CS96-491). San Diego, CA: Department of Computer Science and Engineering, University of California.

[190] Maynard Smith, J. (1982). *Evolution and the theory of games*: Cambridge University Press.

[191] Axelrod, R. M. (1984). *The evolution of cooperation*. New York: Basic Books.

[192] Axelrod, R. M. (1987). Evolving new strategies: the evolution of strategies in the iterated prisoner's dilemma. In L. Davis (Ed.), *Genetic Algorithms and Simulated Annealing*. San Mateo, CA: Morgan Kaufmann Publishers, 32-41.

[193] Hillis, W. D. (1991). Co-evolving parasites improve simulated evolution as an optimization procedure. In C. G. Langton, C. Taylor, J. D. Farmer & S. Rasmussen (Eds.), *Artificial Life II* (Vol. X). Santa Fe Institute Studies in the Sciences of Complexity, New Mexico, USA: Addison-Wesley, 313-324.

[194] Paredis, J. (1995). Artificial coevolution, explorations in artifical life. In *AI Expert Presents*: Miller Freeman Inc.

[195] Pagie, L., & Mitchell, M. (2002). A comparison of evolutionary and coevolutionary search. *International Journal of Computational Intelligence and Applications, 2*(1), 53-69.

[196] Potter, M. A., & De Jong, K. A. (1994). *A cooperative coevolutionary approach to function optimization.* In Y. Davidor, H.-P. Schwefel & R. Männer (Eds.), Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN-III), Jerusalem, Israel, 249-257.

[197] Potter, M. A. (1997). *The design and analysis of a computational model of cooperative coevolution.* Ph.D. dissertation, Computer Science Department, George Mason University, Fairfax, VA.

[198] Luke, S., & Wiegand, R. P. (2002). *When coevolutionary algorithms exhibit evolutionary dynamics.* In A. Barry (Ed.), Proceedings of the Workshop on Understanding Coevolution: Theory and Analysis of Coevolutionary Algorithms (GECCO 2002), New York, 236-241.

[199] Wiegand, R. P., Liles, W. C., & De Jong, K. A. (2001). *An empirical analysis of collaboration methods in cooperative coevolutionary algorithms.* In L. Spector & E. D. Goodman (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), San Francisco, CA, 1235-1242.

[200] Maher, M. L., & Poon, J. (1996). Modelling design exploration as co-evolution. *Microcomputers in Civil Engineering, 11*(3), 195-210.

[201] Maher, M. L. (1994). Creative design using a genetic algorithm. *Computing in Civil Engineering*, 2014-2021.

[202] Maher, M. L., & Poon, J. (1995). *Evolving a fitness landscape for design exploration.* In Proceedings of the International Conference on Evolutionary Computation, Perth, Australia.

[203] Maher, M. L., Poon, J., & Boulanger, S. (1996). Formalising design exploration as co-evolution: a combined gene approach. In J. S. Gero & F. Sudweeks (Eds.), *Advances in Formal Design Methods for CAD*: Chapman & Hall, 1-28.

[204] Maher, M. L., & Wu, P. X. (1998). Creativity through co-evolutionary design. In J. S. Gero, M. L. Maher & F. Sudweeks (Eds.), *Preprints of Computational Models of Creative Design*. Sydney, Australia: Key Center of Design Computing, 244-259.

[205] Poon, J., & Maher, M. L. (1996). Emergent behaviour in co-evolutionary design. In J. S. Gero (Ed.), *Artificial Intelligence in Design '96*: Kluwer Academic Press
[206] Poon, J., & Maher, M. L. (1996). *Co-evolution and emergence in design.* In Proceedings of the Workshop on Evolutionary Systems in Design AID'96.
[207] Poon, J., & Maher, M. L. (1997). Co-evolution and emergence in design. *Artificial Intelligence in Engineering, 11*(3), 319-327.
[208] Nair, P. B., & Keane, A. J. (2002). Coevolutionary architecture for distributed optimization of complex coupled systems. *AIAA Journal, 40*(7), 1434-1443.
[209] Hamda, H., Jouve, F., Lutton, E., Schoenauer, M., & Sebag, M. (2002). Compact unstructured representations for evolutionary topological optimum design. *Applied Intelligence, 16*, 139-155.
[210] Berke, L., & Khot, N. S. (1987). Structural optimization using optimality criteria. In C. A. Mota Soares (Ed.), *Computer Aided Optimal Design: Structural and Mechanical System*. Berlin: Springer-Verlag, 235-269.
[211] Jakiela, M. J., Chapman, C. D., Duda, J., Adewuya, A., & Saitou, K. (2000). Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186*, 339-356.
[212] Schmit, L. A. (1981). Structural synthesis- its genesis and development. *AAAI Journal, 19*(10), 1249-1263.
[213] Lin, C.-Y., & Hajela, P. (1993). *Genetic search strategies in large scale optimization.* In Proceedings of the 34th AIAA/ASCE/ASME/AHS Structural Dynamics and Material Conference, La Jolla, CA, 2437-2447.
[214] Schoenauer, M., & Wu, Z. (1993). *Discrete optimal design of structures by genetic algorithms.* In B. e. al. (Ed.), Proceedings of the Conference Nationale sur le Calcul de Structures, Hermes, 833-842.
[215] Anagnostou, G., Ronquist, E., & Patera, A. (1992). A computational procedure for part design. *Computer Methods in Applied Mechanics and Engineering, 97*, 33-48.
[216] Hajela, P., & Lee, E. (1995). Genetic algorithms in truss topological optimization. *Journal of Solids and Structures, 32*(22), 3341-3357.
[217] Jensen, E. D. (1992). *Topological structural design using genetic algorithms.* Ph.D. DissertationPurdue University, Lafayette, IN.
[218] Chapman, C. D., Saitou, K., & Jakiela, M. J. (1994). Genetic algorithm as an approach to configuration and topology design. *Journal of Mechanical Design, 116*, 1005-1012.
[219] Bendsoe, M. P., & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering, 71*, 197-224.
[220] Xie, Y. M., & Steven, G. P. (1992). *Shape and layout optimization via an evolutionary procedure.* In Proceedings of the International Conference on Computational Engineering Science, Hong Kong University of Science and Technology, Hong Kong.
[221] Sandgren, E., Jensen, E. D., & Welton, J. (1990). Topological design of structural components using genetic optimization methods. In *Sensitivity Analysis and Optimization with Numerical Methods, AMD-vol. 115, Proceedings of the Winter Annual Meeting of the American Society of Mechanical Engineers*. Dallas, TX, 31-43.
[222] Periaux, J., & Winter, G. (Eds.). (1995). *Genetic algorithms in engineering and computer science*. Chichester, UK: John Wiley.
[223] Schoenauer, M. (1996). *Shape representations and evolution schemes.* In L. J. Fogel, P. J. Angeline & T. Bäck (Eds.), Proceedings of the Fifth Annual Conference on Evolutionary Programming, San Diego, CA, USA.
[224] Topping, B. H. V. (1983). Shape optimization of skeletal structures: a review. *Journal of Structural Engineering, 109*, 1933-1951.
[225] Shankar, N., & Hajela, P. (1991). Heuristics driven strategies for near-optimal structural topology development. In B. H. V. Topping (Ed.), *Artificial Intelligence and Structural Engineering*. Oxford, UK: Civil-Comp Press, 219-226.
[226] Hajela, P., Lee, E., & Lin, C.-Y. (1993). Genetic algorithms in structural topology optimization. In M. P. Bendsoe & C. A. Mota Soares (Eds.), *Topology Design of Structures*, 117-133.
[227] Grierson, D. E., & Pak, W. (1993). Discrete optimal design using a genetic algorithm. In M. P. Bendsoe & C. A. Mota Soares (Eds.), *Topology Design of Structures*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 89-102.

[228] Bramlette, M. F., & Bouchard, E. E. (1991). Genetic algorithms in parametric design of aircraft. In L. Davis (Ed.), *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 109-123.
[229] 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.
[230] Bohnenberger, O., Hesser, J., & Männer, R. (1995). *Automatic design of truss structures using evolutionary algorithms.* In Proceedings of the Second IEEE International Conference on Evolutionary Computation (ICEC'95), Perth, Australia, 143-149.
[231] Rajan, S. D. (1995). Sizing, shape, and topology design optimization of trusses using genetic algorithm. *Journal of Structural Engineering, 121*, 1480-1487.
[232] Nakanishi, Y., & Nakagiri, S. (1996). Optimization of frame topology using boundary cycle and genetic algorithms. *JSME International Journal, Series A, 39*, 279-285.
[233] Nakanishi, Y., & Nakagiri, S. (1997). Structural optimization under topological constraint represented by homology groups. *JSME International Journal, Series A*, *40*, 219-227.
[234] Rajeev, S., & Krishnamoorthy, C. S. (1997). Genetic algorithms-based methodologies for design optimization of trusses. *Journal of Structural Engineering, 123*(3), 350-358.
[235] Murawski, K., Arciszewski, T., & De Jong, K. A. (2001). Evolutionary computation in structural design. *Journal of Engineering with Computers, 16*, 275-286.
[236] Soh, C. K., & Yang, Y. (2001). Genetic programming-based approach for structural optimization. *Journal of Computing in Civil Engineering, 31*, 31-37.
[237] Yang, Y., & Soh, C. K. (2002). Automated optimum design of structures using genetic programming. *Computers & Structures, 80*(18-19), 1537-1546.
[238] Azid, I. A., Kwan, A. S. K., & Seetharamm, K. N. (2002). An evolutionary approach for layout optimization of a three-dimensional truss. *Structural and Multidisciplinary Optimization, 24*(4), 333–337.
[239] Rozvany, G. I. N., Bendsoe, M. P., & Kirsch, U. (1995). Layout optimization of structures. *Applied Mechanics Reviews, 48*(2), 41-120.
[240] Bendsoe, M. P., & Sigmund, O. (2002). *Topology optimization: theory, methods and applications*: Springer-Verlag.
[241] Xie, Y. M., & Steven, G. P. (1997). *Evolutionary structural optimization*. Berlin Heidelberg New York: Springer-Verlag.
[242] Pironneau, O. (1984). *Optimal shape design for elliptic systems*: Springer-Verlag.
[243] Bennet, J. A., & Botkin, M. E. (Eds.). (1986). *The optimum shape: automated structural design*. New York, London: Plenum Press.
[244] Haslinger, J., & Neittaanmaki, P. (1996). *Finite element approximation for optimal shape design material and topology design*. Chichester, UK: John Wiley & Sons.
[245] Sokolowski, J., & Zolesio, J.-P. (1992). *Introduction to shape optimization. Shape sensitivity analysis*: Springer-Verlag.
[246] Allaire, G., Bonnetier, E., Francfort, G., & Jouve, F. (1997). Shape optimization by the homogenization method. *Nuemerische Mathematik, 76*, 27-68.
[247] Jenkins, W. M. (1991). Towards structural optimization via the genetic algorithm. *Computers & Structures, 40*(5), 1321-1327.
[248] Jenkins, W. M. (1991). Structural optimization with the genetic algorithm. *Structural Engineer, 69*(24), 418-422.
[249] Richards, R., & Sheppard, S. D. (1992). *Learning classifier systems in design optimization.* In Proceedings of the 1992 Design Theory and Methodology Conference, Scottsdale, Arizona, 179-186.
[250] Watabe, H., & Okino, N. (1993). *A study of genetic shape design.* In S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, USA, 445-451.
[251] Kita, E., & Tanie, H. (1998). GA-based topology optimization of continuum structures. In G. P. Steven, O. M. Querin, H. Guan & Y. M. Xie (Eds.), *Structural Optimization (Proceedings of the Australasian Conference on Structural Optimization)*. Victoria: Oxbridge Press, 87-94.
[252] Kita, E., & Tanie, H. (1999). Topology and shape optimization of continuum structures using GA and BEM. *Structural Optimization, 17*(2/3), 130-139.
[253] Annicchiarico, W., & Cerrolaza, M. (1999). Finite elements, genetic algorithms and B-splines: a combined technique for shape optimization. *Finite Elements in Analysis and Design, 33*, 125-141.

[254] Cerrolaza, M., & Annicchiarico, W. (1999). Genetic algorithms in shape optimization: finite and boundary element applications. In K. Miettinen, M. M. Makela, P. Neittaanmaki & J. Periaux (Eds.), *Evolutionary algorithms in engineering and computer science*. Chichester, England: John Wiley & Sons

[255] Wibowo, F. X. N., & Besari, M. S. (1998). Genetic algorithms in shape optimization of oval axially symmetrical shells. In G. P. Steven, O. M. Querin, H. Guan & Y. M. Xie (Eds.), *Structural Optimization (Proceedings of the Australasian Conference on Structural Optimization)*. Victoria: Oxbridge Press, 103-111.

[256] 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.

[257] 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.

[258] Pedersen, P. (1987). Optimal joint positions for space structures. *Journal of Structural Engineering, 99*(10), 2459-2477.

[259] Vanderplaats, G. N. (1975). *Design of structures for optimum geometry* (No. TMX-62-462): NASA.

[260] Grierson, D. E., & Pak, W. (1993). Optimal sizing, geometrical and topological design using a genetic algorithm. *Structural Optimization, 6*, 151-159.

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

[262] Keane, A. J., & Brown, S. M. (1996). *The design of a satellite boom with enhanced vibration performance using genetic algorithm techniques.* In I. C. Parmee (Ed.), Proceedings of the Conference on Adaptive Computing in Engineering Design and Control 96, Plymouth, UK, 107-113.

[263] Kawohl, B., Pironneau, O., Tartar, L., & Zolesio, J.-P. (Eds.). (2000). *Optimal shape design*. Berlin New York Heidelberg: Springer-Verlag.

[264] Allaire, G., & Henrot, A. (2001). On some recent advances in shape optimization. *Comptes Rendus de l Academie des Sciences Series IIB Mechanics, 329*(5), 383-396.

[265] Nishino, F., & Duggal, R. (1990). Shape optimum design of trusses under multiple loading. *Journal of Solids and Structures, 19*, 17-27.

[266] Arora, J. S. (1989). *Introduction to optimum design*: McGraw Hill.

[267] Hajela, P. (1990). Genetic search - an approach to the nonconvex optimization problem. *AIAA Journal, 26*, 1205-1212.

[268] Hajela, P. (1992). Genetic algorithms in automated structural synthesis. In B. H. V. Topping (Ed.), *Optimization and Artificial Intelligence in Civil and Structural Engineering* (Vol. 1): Kluwer Academic Press

[269] Deb, K. (1991). Optimal design of a welded beam via genetic algorithms. *AIAA Journal, 29*, 2013-2015.

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

[271] Jarmai, 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*, 54–61.

[272] Kicinger, R., Arciszewski, T., & De Jong, K. A. (2004). Morphogenic evolutionary design: cellular automata representations in topological structural design. In I. C. Parmee (Ed.), *Adaptive Computing in Design and Manufacture VI*. London, UK: Springer-Verlag, 25-38.

[273] Kicinger, R., Arciszewski, T., & De Jong, K. A. (2004). *Morphogenesis and structural design: cellular automata representations of steel structures in tall buildings.* In Proceedings of the Congress on Evolutionary Computation (CEC'2004), Portland, Oregon, 411-418.

[274] Rajeev, S., & Krishnamoorthy, C. S. (1992). Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering, 118*(5), 1233-1250.

[275] Sandgren, E., & Jensen, E. D. (1992). *Automotive structural design employing a genetic optimization algorithm.* In Proceedings of the SAE International Congress and Exposition, Detroit, Michigan, SAE Technical Paper #920772.

[276] Adeli, H., & Cheng, N. T. (1993). Integrated genetic algorithm for optimization of space structures. *Journal of Aerospace Engineering, 6*(4), 315-328.

[277] Chapman, C. D., Saitou, K., & Jakiela, M. J. (1993). *Genetic algorithms as an approach to configuration and topology design.* In Proceedings of the ASME 19th Design Automation Conference: Advances in Design Automation, New York, 485-498.

[278] Sakamoto, J., & Oda, J. (1993). *Technique for optimal layout design for truss structures using genetic algorithms.* In Proceedings of the 34th AIAA/ASCE/ASME/AHS Structural Dynamics and Material Conference AIAA/ASME Adaptive Structures Forum, New York, NY, 2402-2408.

[279] Coello Coello, C. A., Rudnick, M., & Christiansen, A. D. (1994). *Using genetic algorithms for optimal design of trusses.* In Proceedings of the Sixth International Conference on Tools with Artificial Intelligence (ICTAI '94), New Orleans, Louisiana, USA, 88-94.

[280] Keane, A. J. (1994). *Experiences with optimizers in structural design.* In I. C. Parmee (Ed.), Proceedings of the First International Conference on Adaptive Computing in Engineering Design and Control, Plymouth, UK, 14-27.

[281] Ohsaki, M. (1995). Genetic algorithms for topology optimization of trusses. *Computers & Structures, 57*(2), 219-225.

[282] Ramasamy, J. V., & Rajasekaran, S. (1996). Artificial neural network and genetic algorithm for the design optimizaton of industrial roofs – a comparison. *Computers & Structures, 58*(4), 747-755.

[283] Cheng, F. Y., & Li, D. (1997). Multi-objective optimization design with Pareto genetic algorithm. *Journal of Structural Engineering, 123*(9), 1252-1261.

[284] Parmee, I. C., Vekeria, H. D., & Bilchev, G. (1997). The role of evolutionary and adaptive search during whole system, constrained and detailed design optimization. *Engineering Optimization, 29*, 151-176.

[285] Yang, J., & Soh, C. K. (1997). Structural optimization by genetic algorithms with tournament selection. *Journal of Computing in Civil Engineering, 11*(3), 195-200.

[286] Jenkins, W. M. (1997). On the application of natural algorithms to structural design optimization. *Engineering structures, 19*(4), 302-308.

[287] de Barros Leite, J. P., & Topping, B. H. V. (1998). Improved genetic operators for structural engineering optimization. *Advances in Engineering Software, 29*(7-9), 529-562.

[288] Camp, C. V., Pezeshk, S., & Cao, G. (1998). Optimized design of two-dimensional structures using a genetic algorithm. *Journal of Structural Engineering, 124*(5), 551-559.

[289] Chen, S.-Y., & Rajan, S. D. (1998). Improving the efficiency of genetic algorithms for frame designs. *Engineering Optimization, 30*, 281-307.

[290] Nair, P. B., Keane, A. J., & Shimpi, R. P. (1998). *Combining approximation concepts with genetic algorithm-based structural optimization procedures.* In Proceedings of the 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, 1741-1751.

[291] Ohmori, H., & Kito, N. (1998). Structural optimization of truss topology by genetic algorithms. *Journal of Theoretical and Applied Mechanics, 47*, 331-340.

[292] Hajela, P., Lee, E., & Cho, H. K. (1998). Genetic algorithms in topologic design of grillage structures. *Computer-Aided Civil and Infrastructure Engineering, 13*(1), 13-22.

[293] Soh, C. K., & Yang, J. (1998). Optimal layout of bridge trusses by genetic algorithms. *Computer-Aided Civil and Infrastructure Engineering, 13*(4), 247-254.

[294] Shrestha, S. M., & Ghaboussi, J. (1998). Evolution of optimum structural shapes using genetic algorithm. *Journal of Structural Engineering, 124*(11), 1331-1338.

[295] Topping, B. H. V., & de Barros Leite, J. P. (1998). Parallel genetic models for structural optimization. *Engineering Optimization, 31*(1), 65-99.

[296] Pezeshk, S., Camp, C. V., & Chen, D. (2000). Design of framed structures by genetic optimization. *Journal of Structural Engineering, 126*(3), 382-388.

[297] Greiner, D., Winter, G., & Emperador, J. M. (2001). Optimizing frame structures by different strategies of genetic algorithms. *Finite Elements in Analysis and Design, 37*, 381-402.

[298] Hajela, P., & Kim, B. (2001). On the use of energy minimization for CA based analysis in elasticity. *Structural and Multidisciplinary Optimization, 23*(1), 24-33.

[299] Deb, K., & Gulati, S. (2001). Design of truss-structures for minimum weight using genetic algorithms. *Finite Elements in Analysis and Design, 37*(5), 447-465.

[300] Sarma, K. C., & Adeli, H. (2001). Bilevel parallel genetic algorithms for optimization of large steel structures. *Computer-Aided Civil and Infrastructure Engineering, 16*, 295-304.

[301] Dimou, C. K., & Koumousis, V. K. (2003). Genetic algorithms in competitive environments. *Journal of Computing in Civil Engineering, 17*(3), 142-149.

[302] Pullmann, T., Skolicki, Z., Freischlad, M., Arciszewski, T., De Jong, K. A., & Schnellenbach-Held, M. (2003). *Structural design of reinforced concrete tall buildings: evolutionary computation approach using fuzzy sets.* In O. Ciftcioglu & E. Dado (Eds.), Proceedings of the 10th International Workshop of the European Group for Intelligent Computing in Engineering (EG-ICE), Delft, The Netherlands.

[303] Kicinger, R., Arciszewski, T., & De Jong, K. A. (2004). *Distributed evolutionary design: island-model based optimization of steel skeleton structures in tall buildings.* In K. Beucke, B. Firmenich, D. Donath, R. Fruchter & K. Roddis (Eds.), Proceedings of the 10th International Conference on Computing in Civil and Building Engineering (ICCCBE-X), Weimar, Germany, 190.

[304] Kicinger, R., & Arciszewski, T. (2004). *Multiobjective evolutionary design of steel structures in tall buildings.* In Proceedings of the AIAA 1st Intelligent Systems Technical Conference, Chicago, Illinois.

Table 4: Chronological classification of applications of EC in structural engineering.

Reference	Domain	Problem	Represen-tation	EA used	Fitness function	Constraint-handling method	Comments
[32]	Shape optimization	Locations of joints in truss systems	Fixed-length, real-valued vectors	ES	Single objective, weight minimization	N/A	A hybrid optimization strategy formed by ES and linear programming
[33]	Sizing optimization	Planar frame under earthquake loading	Fixed-length, real-valued vectors	ES	Single objective, weight minimization	N/A	
[102]	Sizing optimization	Cross-sections in planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	
[267]	Sizing optimization	Cross-sections in planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[221]	Continuum TOD	Planar cantilever plates	Fixed-length, 2D binary arrays (bitarrays)	GA	Single objective, weight minimization	N/A	
[269]	Sizing optimization	Welded beams	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[247] and [248]	Continuum SO	Shape of structural members	Fixed-length, 2D binary arrays (bitarrays)	GA	Single objective, weight minimization	Penalty function	
[225]	Discrete TOD	Planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[268]	Sizing optimization	Cross-sections in planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	

[173]	Sizing optimization	Planar truss systems	Fixed-length, binary strings	GA	Multiobjective, min-max approach	N/A	
[217]	Continuum TOD	Planar cantilever plates	Fixed-length, 2D binary arrays (bitarrays)	GA	Single objective, weight minimization	N/A	
[274]	Sizing optimization	Cross-sections in planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[249]	Continuum SO	Shape of structural members	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[275]	Continuum TOD	Planar cantilever plates	Fixed-length, 2D binary arrays (bitarrays)	GA	Single objective, weight minimization	N/A	
[276]	Sizing optimization	Spatial truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	
[277]	Continuum TOD	Planar cantilever plates	Fixed-length, 2D binary arrays (bitarrays)	GA	Single objective, weight minimization	N/A	
[213]	Sizing optimization	Cross-sections in planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[226]	Discrete TOD	Planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	
[227]	Discrete TOD, SO, and sizing optimization	Planar frame systems	Fixed-length binary strings	GA	Single objective, weight minimization	N/A	
[134]	Sizing optimization	Planar truss systems	Fixed-length, real valued vectors	GA	Single objective, weight minimization	Behavioral memory	
[250]	Continuum SO	Shape of structural members	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	

[278]	Discrete TOD and sizing optimization	Planar truss systems	Fixed-length, binary strings	GA combined with optimality criteria method	Single objective, weight minimization	N/A	GA optimized the layout of the truss and optimality criteria method optimized cross-sections
[140]	Sizing optimization	Spatial truss systems	Fixed-length binary strings	GA	Single objective, weight minimization	Penalty function and augmented Lagrangian	Augmented Lagrangian transforms the constrained problem to an unconstrained one
[218]	Continuum TOD and SO	Planar cantilever plate	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	
[279]	Sizing optimization	Planar and spatial truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	
[229]	Discrete TOD and SO	Planar steel truss roofs	Fixed-length, binary strings	GA combined with logic programming	Single objective, weight minimization	N/A	
[280]	Discrete SO	Planar truss system (satellite boom)	Fixed-length, binary strings	GA	Single objective, minimization of vibration	Penalty function	
[230]	Discrete TOD	Pylon structures	Fixed-length, binary strings	GA and ES	Single objective, weight minimization	N/A	GA produce only topology and ES further optimizes the structures
[231]	Discrete TOD , SO and sizing optimization	Spatial truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	
[216]	Discrete TOD	Planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	Ground structure approach
[281]	Discrete TOD	Planar truss systems	Fixed-length, binary strings	GA	Single objective, total cost	Penalty function	

[148] and [147]	Discrete TOD	Planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Immune network	
[262]	Discrete SO	Spatial truss systems (satellite boom)	Fixed-length, binary strings	GA	Single objective, minimization of vibration	N/A	
[261]	Discrete SO	Planar and spatial truss systems	Fixed-length, binary strings	GA with fuzzy logic	Single objective, weight minimization	Fuzzy logic	
[282]	Discrete TOD and sizing optimization	Planar truss systems	Fixed-length, binary strings	GA combined with neural network	Single objective, weight minimization	Penalty function	
[232] and [233]	Discrete TOD	Planar frame and panel structures	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[283]	Sizing optimization	Planar truss systems	Fixed-length, binary strings	Pareto GA (MOGA)	Multiobjective, with 2 or 3 objectives	Fuzzy penalty function	
[284] and subsequent papers	Discrete TOD, SO, and sizing optimization	Various problems considered	Various encodings (binary, real, etc.)	Various kinds of EAs (GA, cluster oriented GAs, CHC, VEGA, ES)	Single and multiobjective approaches	Various constraint-handling methods	Proposed the “whole system design” in which various EAs are applied at different stages of design process
[285]	Discrete SO	Planar truss systems	Fixed-length, binary strings	GA using tournament selection	Single objective, weight minimization	N/A	
[234]	Discrete TOD, SO, and sizing optimization	Planar truss structures	Variable-length, binary strings	GA	Single objective, weight minimization	N/A	

[286]	Discrete SO and sizing optimization	Planar multistory frame structure with truss-supported hangers	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	Adaptation of mutation and crossover rates
[287]	Discrete TOD and sizing optimization	Welded beam, planar truss systems, and prestressed I-sections	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	Various modifications of genetic operators
[288]	Sizing optimization	Planar truss and frame structures	Fixed-length, binary strings	GA	Single objective, various fitness functions	Penalty function	Various penalty functions studied
[289]	Discrete TOD, SO, and sizing optimization	Planar frame systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Adaptive penalty function	Specialized one-point crossover operator using so-called association string
[290]	Sizing optimization	Planar truss system	N/A	GA combined with approximation models	Single objective, weight minimization	Penalty function	Adaptive selection operator

[291]	Discrete TOD	Planar and spatial truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	Binary representation encodes triangles rather than single structural members
[292]	Discrete TOD	Planar and spatial grillage structures	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	Two-level GA
[293]	Discrete TOD, SO, and sizing optimization	Planar bridge trusses	Fixed-length, binary strings	GA utilizing domain knowledge	Single objective, weight minimization	Penalty function	Domain knowledge encoded in so-called cognitive topologic patterns
[294]	Discrete TOD	Planar truss systems	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	
[295]	Sizing optimization	Cable-stayed bridge	Fixed-length, binary strings	Parallel GA	Single objective	N/A	Various strategies and topologies of parallel GAs discussed
[255]	Continuum SO	oval axially symmetric shells	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	

[251] and [252]	Continuum SO	Planar structures	Fixed-length, binary strings	GA	Single objective, weight minimization	N/A	Shape defined by B-spline functions
[253]	Continuum SO	Planar structures	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty function	Shape defined by B-spline functions
[86]	Continuum structural elasticity analysis	Planar structures	Cellular automata	GA	Single objective, strain energy minimization	Penalty function	
[175]	Sizing optimization	Planar and spatial truss systems	Fixed-length, binary strings	GA with min-max strategy	Multiobjective, weight, displacement, and stress minimization	Penalty function (death penalty)	
[296]	Sizing optimization	Planar multi-story frame systems	Fixed-length, binary strings	GA	Single objective, weight minimization	Penalty functions	

[6]	Discrete TOD and sizing optimization	Steel skeleton structures in tall buildings	Fixed-length, integer encodings	Parallel unified EA (island-model with migrations)	Single objective, weight minimization	Penalty function (death penalty), repair mechanisms	
[257]	Continuum SO	2D spanner head and flange webbing	Fixed-length, binary strings	GA	Single objective, weight and deflection minimization	None	
[297]	Sizing optimization	Planar frame structures	Fixed length, binary strings	GA and CHC for single objective optimization, and NSGA for multiobjective optimization	Single objective (weight minimization), and multiobjective (weight minimization and number of member cross-sections minimization)	Penalty functions	
[235]	Discrete TOD and sizing optimization	Steel skeleton structures in tall buildings	Fixed-length, integer encodings	ES	Single objective, weight minimization	Penalty function (death penalty), repair mechanisms	
[256]	Continuum SO	3D cantilever plate with circular hole	Fixed-length, binary strings	GA	Single objective, minimization of volume	Penalty function	

[298]	Continuum structural elasticity analysis	Planar structures	Binary and real encodings and cellular automata	GA	Single objective, strain energy minimization	Penalty function	
[111]	Sizing optimization	Planar truss systems and frame	Fixed-length, binary strings	GA	Single objective, weight minimization	Adaptive penalty function	
[299]	Discrete TOD , SO and sizing optimization	Planar and spatial truss systems	Fixed-length, real valued vectors	GA	Single objective, weight minimization	Penalty function	
[184]	Continuum SO	Planar plate structures	Fixed-length, binary strings	NSGA-II combined with hill climber	Multiobjective, weight and displacement minimization		
[300]	Sizing optimization	Spatial multistory frame structures	Fixed-length, binary strings	Parallel GA with migrations (island-model)	Single and multiobjective		Bi-level GA: - level 1 - single objective weight minimization, level 2 - multiobjective cost optimization
[237]	Discrete TOD	Planar truss systems	Variable-length, parse trees	GP	Single objective, weight minimization	Penalty functions	

[208]	Sizing optimization	Planar truss systems	Fixed-length, binary strings	Cooperative coevolutionary algorithm (CCEA)	Single objective, weight minimization	Penalty functions	
[238]	Discrete TOD	Planar and spatial truss systems	Fixed-length, real valued vectors	GA	Single objective, weight minimization	Penalty functions	
[209]	Continuum TOD	Planar and spatial cantilever plates	Variable-length, Voronoi-based, and fractal-based	GA	Single objective, weight minimization	Penalty function	
[183]	Continuum TOD	Planar cantilever plate	Variable-length, Voronoi-based	NSGA-II	Multiobjective, weight and displacement minimization		
[130]	Discrete TOD	Steel skeleton structures in tall buildings	Fixed-length, integer representations	ES	Single objective, weight minimization	Penalty function (death penalty), repair mechanisms	
[301]	Sizing optimization	Planar truss systems	Fixed-length, binary strings	Parallel GA	Single objective for individuals in each population – total cost	Penalty function	(coevolving populations without migrations competing for limited resources)

[302]	Discrete TOD	Reinforced concrete tall buildings	Fixed-length, integer strings	Unified EA and fuzzy sets	Single objective, total cost	Fuzzy logic	
[272]	Discrete TOD and Sizing optimization	Wind bracing systems in tall buildings	Generative representations based on cellular automata (1D and 2D)	ES	Single objective, the total weight	Penalty function (death penalty), repair mechanisms	
[273]	Discrete TOD and Sizing optimization	Steel structural systems in tall buildings	Generative representations based on 1D cellular automata	ES	Single objective, the total weight	Penalty function (death penalty), repair mechanisms	
[303]	Discrete TOD and sizing optimization	Steel structural systems in tall buildings	Fixed-length, integer representations	Distributed EA (island-model)	Single objective, the total weight	Penalty function (death penalty), repair mechanisms	
[304]	Discrete TOD and sizing optimization	Steel structural systems in tall buildings	Fixed-length, integer representations	ES	Multiobjective (aggregate function), the total weight and the maximum horizontal displacement	Penalty function (death penalty), repair mechanisms	