Knowledge Interaction With Genetic Programming in Mechatronic Systems Design Using Bond Graphs

Jiachuan Wang, Zhun Fan, Janis P. Terpenny, and Erik D. Goodman

***Abstract*—This paper describes a unified network synthesis approach for the conceptual stage of mechatronic systems design using bond graphs. It facilitates knowledge interaction with evolutionary computation significantly by encoding the structure of a bond graph in a genetic programming tree representation. On the one hand, since bond graphs provide a succinct set of basic design primitives for mechatronic systems modeling, it is possible to extract useful modular design knowledge discovered during the evolutionary process for design creativity and reusability. On the other hand, design knowledge gained from experience can be incorporated into the evolutionary process to improve the topologically open-ended search capability of genetic programming for enhanced search efficiency and design feasibility. This integrated knowledge-based design approach is demonstrated in a quarter-car suspension control system synthesis and a MEMS bandpass filter design application.**

***Index Terms*—Bond graphs, controller synthesis, genetic programming, knowledge interaction, mechatronics, MEMS filter design.**

I. Introduction

EVOLUTIONARY algorithms (EAs) are powerful but general algorithms relying on few assumptions about the search space. These characteristics provide both the opportunity and necessity to include knowledge to enhance the efficiency and applicability of EAs in various domain applications. Much research has been carried out in this area, to represent and incorporate knowledge in, and to extract knowledge from evolutionary algorithms in a variety of ways. Examples include representing knowledge in fuzzy logic [1], [2] and neural networks [3], incorporating problem-specific knowledge [4], [5] and preferences [6]–[8] into evolutionary algorithms, as well as extracting knowledge during the evolutionary process [9]. The inclusion of knowledge in evolutionary computation is also tightly related to human-computer interaction [10], since knowledge is always related to human understanding.

While many applications are focused on one aspect of knowledge deployment in the evolutionary process, it is desirable to develop a unified framework to address knowledge interaction with evolutionary computation in an integrated and comprehensive manner. This can facilitate the adoption of evolutionary computation in real-world industrial applications, since better insight into the problem domain is gained through iterative human-comprehensible knowledge discovery and reuse from this "black box" optimization technique. In addition, the knowledge acquired during the evolutionary process may assist the human analyst in refining problem objectives and identifying better directions for future investigation.

The goal of this work is to develop a unified framework to address knowledge interaction with evolutionary computation as applied to conceptual mechatronic systems design. While there are other representation methods for mechatronics (for example, block diagrams, linear graphs [11], etc.), bond graphs language is particularly suited in this work to model mechatronic systems at the conceptual design stage for its unified representation across various engineering domains (e.g., mechanical, electrical, and control systems, etc.). Bond graphs can be conveniently encoded in a genetic programming tree representation to explore various design configurations and parameterizations. With the open-ended search capability of genetic programming to generate emergent behavior [12], this approach has potential to span a large search space and find better and creative mechatronics design solutions. This work belongs to the challenging computational synthesis research that has gained more attention recently, which seeks to assemble low-level design primitives, or features, to achieve given arbitrary high-level functionality [13]. Previous research works in mechatronics design using bond graphs and genetic programming with detailed explanation of bond graph/genetic programming (BG/GP) encoding and decoding can be found in [14]–[16].

The basic BG/GP approach provides a foundation to test the effectiveness of including knowledge concept on the computational synthesis experimentation. In this paper, it is shown that by structuring the design primitives into modular building blocks inductively and deductively, our approach facilitates knowledge interaction with evolutionary algorithms significantly. Since bond graphs provide a succinct set of basic building blocks for mechatronic systems modeling, it is possible to extract useful modular design knowledge discovered during the evolutionary process for design creativity and reusability. Moreover, design knowledge gained from experience can be incorporated into the evolutionary process to improve the topologically open-ended search capability of genetic programming for improved search efficiency and design feasibility.

In the remainder of this paper, an overview of the unified mechatronics evolutionary synthesis architecture is presented in Section II. Based on the overall design approach, a quarter-car suspension control system design and a MEMS bandpass filter

Manuscript received September 1, 2003; revised February 20, 2004. This work was supported in part by the National Science Foundation under Grants EEC-0332058 and DMII-0084934. This paper was recommended by Guest Editor Y. Jin.

J. Wang is with United Technologies Research Center, East Hartford, CT 06108 USA (e-mail: wangj2@utrc.utc.com).

Z. Fan is with the Technical University of Denmark, Lyngby, Denmark (e-mail: zf@mek.dtu.dk).

J. P. Terpenny is with Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: terpenny@vt.edu).

E. D. Goodman is with Michigan State University, East Lansing, MI 48824 USA (e-mail: goodman@egr.msu.edu).

Digital Object Identifier 10.1109/TSMCC.2004.841915

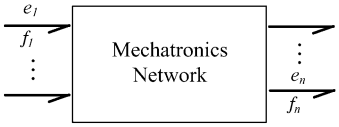


Fig. 1. Mechatronics n-port network.

design are presented in Sections III and IV, respectively, to address the challenge of computational synthesis by extracting, as well as incorporating, knowledge in the evolutionary process. Finally conclusions are provided in Section V, highlighting the need, value, and future work of the proposed approach.

II. Unified Mechatronics Evolutionary Synthesis Architecture

In our approach, mechatronics design is treated as a network synthesis problem using bond graphs, which are combined with genetic programming to evolve alternative mechatronics design solutions based on design performance requirements. This approach integrates active design and passive design by also designing controller schemes using bond graphs. In order to constrain the search space, realizable function sets (as discussed in Section D) are used to establish the problem-specific GP primitives for each design problem.

A. Mechatronics Design as Network Synthesis

The term "mechatronics" probably was first created in 1969 [17]. It is an interdisciplinary field involving the following disciplines [18]:

- mechanical systems (mechanical elements, machines, precision mechanics);
- electronic systems (microelectronics, power electronics, sensor, actuator, and controller technology);
- information technology (system theory, automation, software engineering, artificial intelligence).

Drawing on the analogy with electrical networks and mechanical networks [19], mechatronic systems with power interaction can be modeled as general multiport networks, with n pairs of effort and flow variables $(e_i, f_i), i = 1, \ldots, n$, as shown in Fig. 1. The product of effort and flow variables is an instantaneous power. Each port represents an interface with other ports. When two multiports are connected, power can flow through the connected ports, which are represented by a single line or bond between the multiports. In Fig. 1, the power bonds are represented with half-arrow following the notation of bond graphs to indicate the direction of power flow.

For a linear multiport network, it is possible to define the frequency domain transfer matrices from effort to flow variables (impedance Z), flow to effort variables (admittance Y), or mixed immittance. Impedance and admittance are defined as follows:

$$\text{impedance } z(s) = \frac{\text{effort}}{\text{flow}} = \frac{e(s)}{f(s)} \tag{1}$$

$$\text{admittance } y(s) = \frac{\text{flow}}{\text{effort}} = \frac{f(s)}{e(s)}. \tag{2}$$

For a two-port network, the impedance matrix is governed by

$$\begin{bmatrix} e_1(s) \\ e_2(s) \end{bmatrix} = \begin{bmatrix} z_{11}(s) & z_{12}(s) \\ z_{21}(s) & z_{22}(s) \end{bmatrix} \begin{bmatrix} f_1(s) \\ f_2(s) \end{bmatrix} = Z \begin{bmatrix} f_1(s) \\ f_2(s) \end{bmatrix}. \tag{3}$$

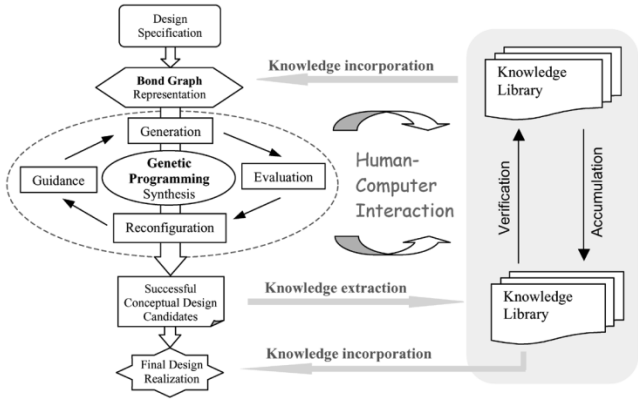


Fig. 2. Overall mechatronics synthesis procedure.

From network synthesis theory [20], an important theorem states:

Theorem 1: Consider a network with impedance $Z(s)$. The network is passive if and only if $Z(s)$ is positive real.

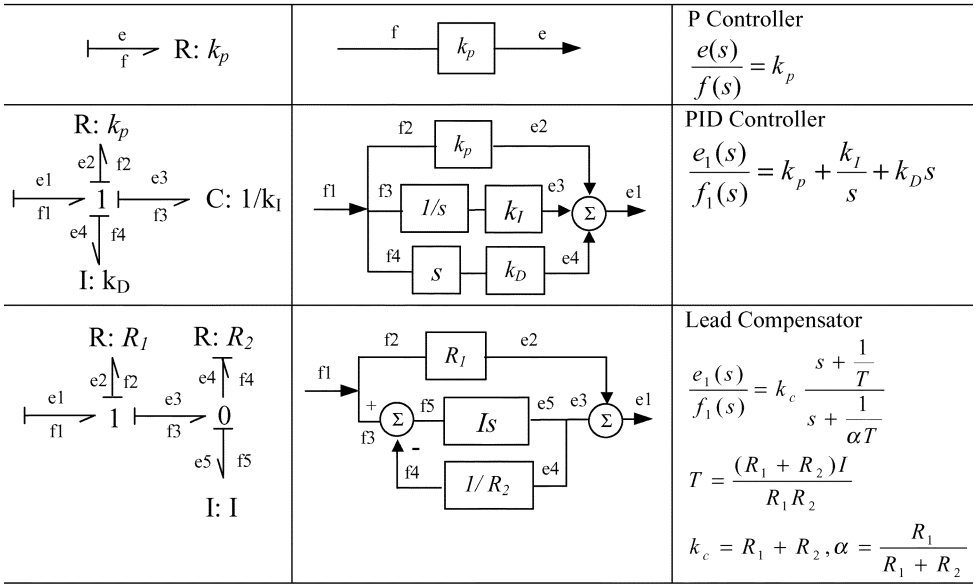
This theorem also holds with $Z(s)$ replaced by admittance matrix $Y(s)$ or a mixed immittance $G(s)$. For network synthesis, the network transfer matrix can be used to specify desired system performance to be achieved. If it is positive real, it can be implemented using passive physical elements, or sometimes an active control structure. If it is not positive real, the system cannot be implemented using passive elements alone. In this situation, extra energy input is necessary for an active implementation.

B. Bond Graphs and Genetic Programming

For conceptual mechatronics design, the bond graphs language serves as a formal schematic modeling tool for multidomain lumped-parameter physical systems [21]. It concisely captures the system's dynamic behavior in terms of energy interaction and conservation. Bond graphs are represented as interconnected components with power flow across their interfaces (ports), which makes it particularly useful as a modeling tool for mechatronics network synthesis. The basic bond graph components are: {One port elements: C—capacitor, I—inertia, R—resistor; Two-port elements: TF—transformer, GY—gyrator; Three-port elements: 0-junction, 1-junction; and Source elements: SE—effort source, SF—flow source}, which can be mapped to different physical realizations in different domains. From a knowledge encapsulation point of view, bond graphs consist of a small set of basic building blocks for conceptual mechatronics design. An introduction to bond graphs can be found in [22].

Originally developed for multidomain dynamic system analysis purposes, bond graphs have also been used as functional and behavioral representation and design synthesis tools [23], [24]. To analyze bond graphs, it is straightforward to calculate the network transfer matrices from a bond graph multiport structure connected by 0-junctions and 1-junctions [25]. To synthesize bond graph structures from a given transfer matrix is an inverse process. In this approach, based on the above mechatronics network definition, design synthesis is achieved by generating alternative bond graph structures interconnected with basic bond graph components from overall network impedance or admittance matrix specifications. The resulting bond graphs

TABLE I
IMPEDANCE-TYPE CONTROLLER SCHEMES IN BOND GRAPHS AND BLOCK DIAGRAMS



structures at the conceptual design level can be associated with physical artifacts in different domains for simulation and physical realization.

The automatic assemblage of basic bond graph components into complex design structures is realized by encoding the structure of a bond graph in a genetic programming tree representation, using a relatively direct mapping from genotype to phenotype, similar as DNA mapping into genetic algorithm string representation in evolutionary biology. This allows a graphical analysis and straightforward exploration of various design configurations and parameterizations by means of evolutionary computation. The overall design procedure is diagrammed in Fig. 2.

The basic BG/GP approach [14], mainly described on the left-hand side of Fig. 2, has demonstrated its potential to iteratively generate, evaluate, and reconfigure design solutions. Since engineering design, as a purposeful knowledge-based activity, needs constant interaction with human designers, improved communication and understanding between cognitive human thoughts (top-down) and automated computation (bottom-up) are expected. Based on previous work, this work focuses on improving the evolutionary design approach through knowledge interaction with genetic programming by extracting and incorporating modular design knowledge during the evolutionary process. Two key concepts related to modular design building blocks are provided in the following two subsections—namely, controller schemes in bond graphs and realizable function sets.

C. Controller Schemes in Bond Graphs

In general, the mechatronics design procedure involves the integration of physical systems design and control systems design, which conventionally require different representation schemes. "Controller design in the physical domain" [26] is proposed as a means to unify control systems design with mechanical systems design. The term physical equivalence states that it is possible to describe the controlled system as an equivalent physical system, provided that ideal actuators and sensors can be placed at any point in the system. This approach facilitates separation of controller representation issues from implementation issues, thus providing guidance at the high-level design stage in selecting the proper overall system structure for a given design task.

As pointed out in the impedance control principle [27], a passive physical system and an active control system can be regarded as two physical systems interacting with each other. Physical systems are treated as being of one of two types: admittances, which accept effort inputs and yield flow outputs; and impedances, which accept flow inputs and yield effort outputs. Real physical systems exist which can be described in one form and not the other. When a passive physical system and an active control system interact, they physically complement each other. For example, if the passive physical system functions as an admittance, i.e., accepting effort and producing a flow response, as in the automotive suspension system case, the active controller should assume the behavior of an impedance, producing effort responses to flow inputs, such as position control and velocity control.

In this work, controller schemes are represented in bond graphs. Control systems can be classified into two major categories, collocated and noncollocated. Collocation means to physically locate actuators and sensors at the same place such that they are energetically conjugated—for example, to apply force in response to displacement (or velocity) measurement. The use of collocated actuators and sensors typically leads to better stability than use of noncollocated control, because of their robustness with respect to uncertainty [28]. Collocated control in a bond graph representation is an effort-flow one-port pair that includes all sensor, controller and actuator effects. The active effort source is generated by the corresponding flow signal measurement through controller modulation, and vice versa.

To design a controller in the physical domain, the controller can be represented by various combinations of bond graph C, I, and R elements, to represent various control schemes, such as P, PI, PD, PID controller, or lead and lag compensators. Table I

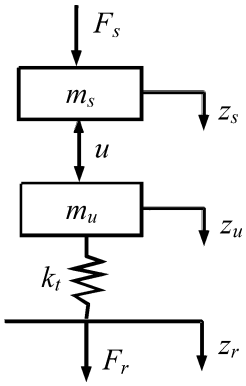


Fig. 3. Physical realization of a PI controller scheme.

TABLE II
BG/GP BASIC FUNCTION SET, A MEMS COMPLEX MODULE FUNCTION SET, AND TERMINAL PRIMITIVE

	Name and description
Basic Function Set	0 – Junction (0) 1 – Junction (1) R Element (R) C Element (C) I Element (I) Arithmetic + (+): add two ERCs Arithmetic – (–): Subtract two ERCs
MEMS filter complex module Function Set	RU: a Resonant Unit CU: a Coupling Unit BU: a Bridging Unit CIR: a special CIR CR: a special CR
Terminal Primitive	Ephemeral Random Constant (ERC)

shows some impedance controller schemes in bond graphs with their corresponding block diagrams and transfer functions. By classifying various controller schemes, we are learning and structuring modular design knowledge into the design library.

When representing controller schemes in bond graphs, the bond graphs themselves are only high-level abstract representations of the controllers; they do not have physical counterparts. A collocated controller scheme in bond graphs is realized by using a sensor and an actuator that are connected by a negative feedback loop. As illustrated in Fig. 3, a force source is acting on a mass with a PI controller generating counteracting force upon it. The PI controller, which consists of one R and one C element in a bond graph representation, is realized by measuring the velocity signal and generating a force proportional to the mass's position and velocity. The force input is realized through an actuator that provides modulated power to the system. Since the direction of the actuating bond is reversed, the modulated force becomes negative thus forming a negative feedback loop.

The inclusion of bond graph controller schemes in mechatronics synthesis unifies design representation of bond graphs for integrated mechatronics systems. It involves coupling a low-power sensing and information-processing system with a high-power system, intended to change the dynamic performance by adding energy to the system. The use of feedback control also reduces the sensitivity of the output to parameter variations and attenuates the effect of disturbances within the bandwidth of the controlled electro-mechanical systems.

D. Realizable Function Sets

The BG/GP approach is quite general for automatic synthesis of mechatronic systems. Using a basic set of building blocks allows construction of many types of unconstrained systems, which may not guide the evolutionary process toward a promising direction in the search space, and may generate nonfeasible solutions. In addition, engineering systems in the real world are often limited by various types of constraints, including shortcomings of existing technologies, which must be integrated into the evolutionary design process. The concept of *realizable function sets* is proposed to expand the basic GP primitive set to include more complex modules. For example, complex modules with the particular property (drawing on domain-specific knowledge) that current technology provides a direct way to physically realize the designs developed with these modules. More stringent constraints on manufacturability can also be imposed in the realizable function sets if needed for a particular application domain.

Definition of realizable function sets for use in genetic programming may affect both the search efficiency and validity of the experimental results. In the work reported here, BG/GP function sets are divided into two groups: 1) basic function set and 2) complex module function sets for various applications. A realizable function set may include functions from both the basic function set and a complex module function set. Table II lists the basic function set, a typical complex module function set for MEMS filter design, and the terminal primitive. The basic function set includes the basic bond graph components and arithmetic operators to calculate design parameters on randomly generated terminal primitives—the so-called ephemeral random constants (ERCs). Primitives in the basic function set are useful to construct more complex structures and allow for free exploration of the search space, while primitives in a complex module function set purport to construct systems using relatively modular and predefined subassemblies, helping to constrain the search space. More details about the MEMS filter design complex module function set are discussed in Section IV.

Different choices of functions from basic function set and complex module function sets lead to different experimental settings and possibly different computational results, thus giving the designer flexibility to determine the tradeoff between search space exploration and exploitation.

Based on the GP tree constructed from the basic function set, complex module function sets and terminal primitive, selection, crossover, and various mutation operations can be applied, for example, to swap junctions and nodes, to shrink subtrees, etc. The Open Beagle framework [29] has been adopted as the evolutionary computation platform for this work. Its architecture follows the principles of object-oriented programming which is well structured and expandable.

The following two sections discuss the overall design approach applied to two design applications, a quarter-car suspension control system design and a MEMS bandpass filter design, reinforcing the effect of knowledge interaction with evolutionary computation on real-world engineering applications.

III. Application to Quarter-Car Suspension Design

A. Problem Description

Suspension systems are important subsystems of most wheeled vehicles. From a system design point of view, there are two main types of disturbances acting on a vehicle, namely road and load disturbances. Road disturbances have the characteristics of large magnitude in low-frequency disturbances (such as hills) and small magnitude in high-frequency disturbances (such as road roughness). Load disturbances include the variations of loads induced by accelerating, braking and cornering. Therefore, a good suspension design is concerned with disturbance rejection from these disturbances to the outputs (e.g., vertical position of vehicle mass) in which performance is evaluated. In general, a suspension system needs to be "soft" to insulate against road disturbances and "hard" to insulate against load disturbances.

A quarter-car iconic model is illustrated in Fig. 4. The sprung mass m_s (kg), consists of the main vehicle body supported by the suspension. The unsprung mass m_u (kg), consists of hub, wheel and tire. The tire is modeled as a spring with stiffness k_t (N/m). z_s, z_u, and z_r are the vertical positions of the sprung mass, the unsprung mass and the road disturbance input, respectively. Force fs is the load force disturbance input. Force u represents any possible suspension force.

From the point of view of a multiport mechatronics network, the quarter-car suspension system can be viewed externally as a two-port network [30], with its corresponding mixed immittance matrix G defined as

$$\begin{bmatrix} F_r \\ \dot{z}_s \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} \dot{z}_r \\ F_s \end{bmatrix}. \tag{4}$$

F_r represents the applied force from the tire to the road. The matrix G can be obtained from the following equations of system motion, together with specified suspension force u:

$$m_s \ddot{z}_s = -u + F_s \tag{5}$$

$$m_u \ddot{z}_u = u + k_t(z_r - z_u). \tag{6}$$

B. Suspension Design With Road Disturbance

First consider the situation when only road disturbance exists. This is a one-degree-of-freedom design problem. It shows in [31] that road disturbance response can be achieved using a dynamic compensator measuring only suspension deflection $(z_s - z_u)$. This can be achieved using a collocated control system. The corresponding bond graph representation is shown in Fig. 5, as an initial embryo structure. $K_I(s)$ is the impedance of the unknown one-port suspension system to be synthesized, which is defined as: $K_I(s) = (u)/(\dot{z}_s - \dot{z}_u) = (u(s))/(s(z_s - z_u))$.

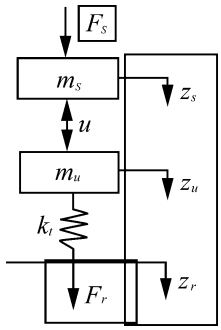


Fig. 4. Quarter-car model in iconic diagram.

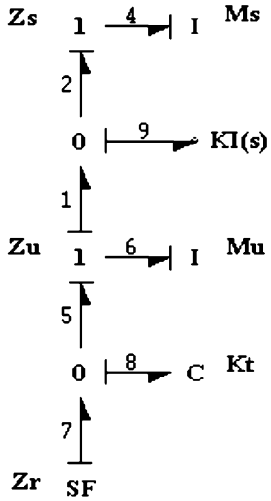


Fig. 5. Suspension design with road disturbance represented in bond graphs.

The experimentation follows the parameter setting for a quarter-car model as in [31], with $m_s = 250$ kg, $m_u = 35$ kg, $k_t = 150 \times 10^3$ N/m. The desired frequency response for road disturbance $H_1(s)$ is obtained by choosing certain suitable parameters in a "double skyhook" configuration: $u = k_s(z_s - z_u) + c_1\dot{z}_s - c_2\dot{z}_u$, which corresponds to a spring between sprung mass and unsprung mass, and sky-hook damper for each sprung mass and unsprung mass. The parameter is specified to be "soft" for road disturbance rejection, $k_s = 1000$ N/m, $c_1 = 4000$ Ns/m, $c_2 = 2000$ Ns/m. $H_1(s)$ is the frequency response specification for $G_{21}(s)$ in the immittance matrix in (4). The desired $H_1(s)$ is calculated as shown in (7) below.

Using suspension impedance $K_I(s), G_{21}(s)$ is obtained as shown in (8) below.

$$H_1(s) = \frac{\dot{z}_s}{\dot{z}_r} = \frac{c_2 k_t s + k_s k_t}{m_s m_u s^4 + (c_1 m_u + c_2 m_s)s^3 + (k_s m_u + k_s m_s + k_t m_s)s^2 + c_1 k_t s + k_s k_t} \tag{7}$$

$$G_{21}(s) = \frac{\dot{z}_s}{\dot{z}_r} = \frac{k_t K_I(s)}{m_s m_u s^3 + (m_s + m_u)K_I(s)s^2 + k_t m_s s + k_t K_I(s)} \tag{8}$$

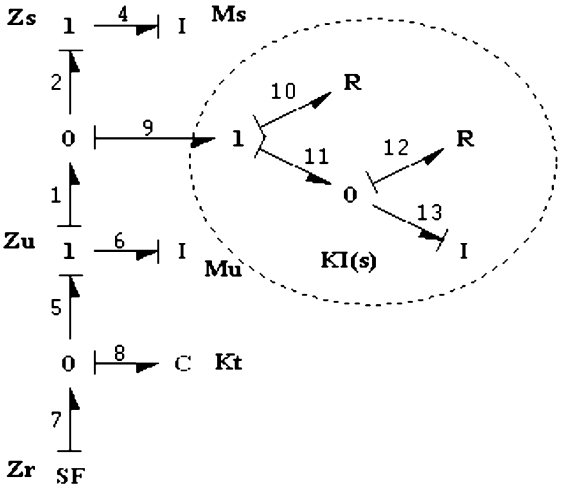


Fig. 6. Best evolved suspension structure for road disturbance.

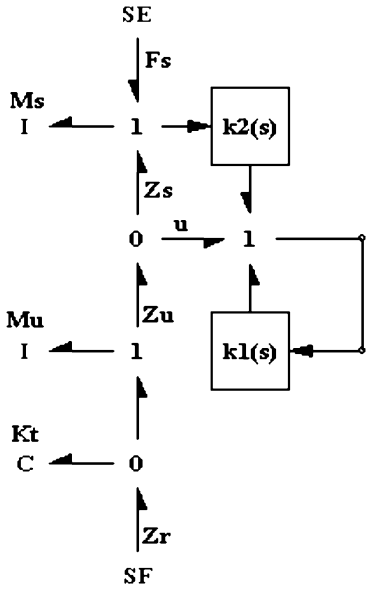


Fig. 7. Quarter-car suspension control with both road and load disturbances.

The design synthesis starts from the desired system specification, on which the mechatronic system depends for its behavior. The evolution of the BG/GP tree is an error-driven function approximation process. The fitness of a GP tree is evaluated by how accurately it approximates the desired frequency domain specification, to minimize $\|\mathrm{dTF}(j\omega)-\mathrm{tTF}(j\omega)\|_2$, where $\mathrm{dTF}(j\omega)$ is the desired frequency response, and $\mathrm{tTF}(j\omega)$ is the theoretical frequency response of an evolved individual bond graph structure to be evaluated.

Using the basic GP function set in Table II, the best run of the BG/GP system yielded the suspension structure shown in Fig. 6, with corresponding impedance $K_I(s) = (1992(s+4.53))/(s+7.31)$.

Theoretically, since $K_I(s)$ is positive-real, it is inherently a passive control law. The bond graph structure for the suspension design may be realized using passive elements in the mechanical domain. However, it is not possible to do so, since a mass with relative velocity between the sprung mass and the unsprung mass does not exist in the mechanical domain. This also shows that a double sky-hook suspension design specification is not physically realizable. It is more justifiable to implement the suspension system as an active controller, which in this case, is a lead compensator, referring to Table I. The lead compensator gives road disturbance response very close to its desired performance as specified in $H_1(s)$. This is an example of knowledge extraction in the evolutionary process similar to the concept of automatically defined functions (ADF) [32], without explicitly defining a function. From this GP experimentation, we gain useful knowledge that the lead compensator increases the damping of the flexible mode. This is reasonable because a lead compensator increases the stability and speed of response of a system. The lead compensator represented as a bond graph can be added to the GP controller complex module function set in the knowledge library for future reuse.

C. Suspension With Both Road and Load Disturbances

When load disturbance is also considered, the suspension system needs to be stiff to loads acting on the sprung mass. This requires in (4), $G_{11}(s)$ and $G_{21}(s)$ be set "soft" to road disturbance rejection while $G_{12}(s)$ and $G_{22}(s)$ be set "hard" to load disturbance rejection. For such design requirements, the matrix G fails to be positive-real, which implies active energy input is necessary for such suspension implementation [30].

There is one degree-of-freedom available for the response to each of the road and load disturbances. They can be determined independently if two suitable measurements are available for feedback (e.g., suspension deflection and sprung mass velocity). The suspension design with two measurements is shown in Fig. 7, with the control law taken to be

$$u = [k_1(s) \quad k_2(s)] \begin{bmatrix} z_s - z_u \\ sz_s \end{bmatrix}$$

where $k_1(s)$ is collocated control, while $k_2(s)$ is noncollocated control.

In order to synthesize controller $k_1(s)$ and $k_2(s)$, desired performance requirements for road and load disturbance rejection are specified. The desired frequency response for road disturbance $H_1(s)$ is the same as in (7). The desired load disturbance frequency response $H_2(s)$ is the frequency response specification for $G_{22}(s)$ in the immittance matrix in (4). It is obtained by choosing certain suitable parameters in another "double sky-hook" configuration: $u = k_s(z_s - z_u) + c_1\dot{z}_s - c_2\dot{z}_u$, with a hard damper and spring configuration with $k_s = 150\,000$ N/m, $c_1 = 12\,000$ Ns/m, $c_2 = 6000$ Ns/m. The desired $H_2(s)$ is calculated as shown in (9) below.

$$H_2(s) = \frac{\dot{z}_s}{F_s} = \frac{(m_u s^2 + c_2 s + k_t + k_s)s}{m_s m_u s^4 + (c_1 m_u + c_2 m_s)s^3 + (k_s m_u + k_s m_s + k_t m_s)s^2 + c_1 k_t s + k_s k_t}. \tag{9}$$

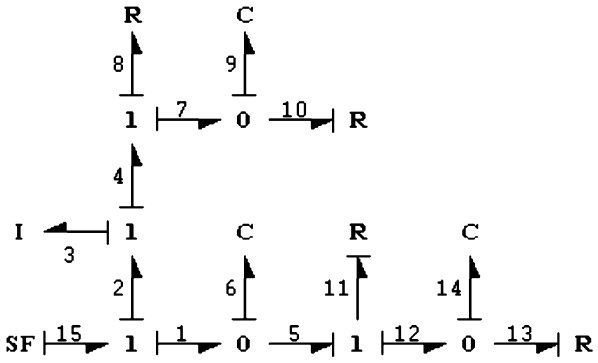


Fig. 8. Controller structure in bond graph form for $k_1(s)$.

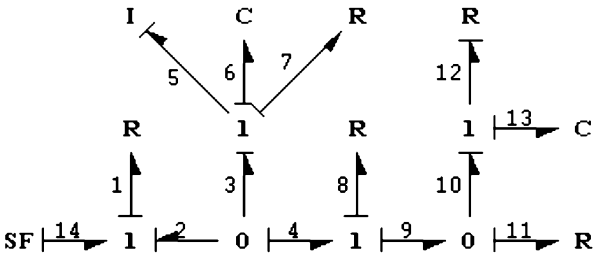


Fig. 9. Controller structure in bond graph form for $k_2(s)$.

Taking the desired road and load disturbance rejection responses $H_1(s)$ and $H_2(s)$ as evaluation criteria, the best run of genetic programming using the basic function set in Table II produced the results shown in Fig. 8 for $k_1(s)$ and Fig. 9 for $k_2(s)$

$$k_1(s) = \frac{-2128s^3 + 46\,680s^2 + 1\,137\,000s + 4\,792\,000}{s^2 + 16.08s + 32.45} = \frac{2128(s + 5.011)(s^2 + 16.93s + 449.4)}{(s + 2.366)(s + 13.71)}$$

$$k_2(s) = \frac{10\,320s^3 - 453\,300s^2 - 40\,260\,000s - 437\,000\,000}{s^3 + 172s^2 + 5799s + 15\,890} = \frac{10\,320(s + 12.04)(s^2 + 31.89s + 3517)}{(s + 3)(s + 41.5)(s + 127.5)}.$$

The degree of a system can be determined by counting independent storage elements present in the bond graph. The controllers obtained here are of lower order than the controllers obtained in [31]. This shows that by applying genetic programming to evolve bond graph control structures, it is potential to discover good control strategies that may not be obtained through conventional methods.

Fig. 10 shows the simulation results as in MATLAB Bode diagrams comparing desired responses (solid lines) with actual responses (dashed lines) realized by active suspension control evolved from evolutionary computation. The left-hand side shows the road disturbance rejection responses, and the right-hand side shows the load disturbance rejection responses. It demonstrates that the actual responses approximate the desired responses very well.

In summary, a passive suspension system has the ability to store energy via a spring and to dissipate it via a damper. Its parameters are generally fixed, being chosen to achieve a certain level of compromise between road holding and load carrying.

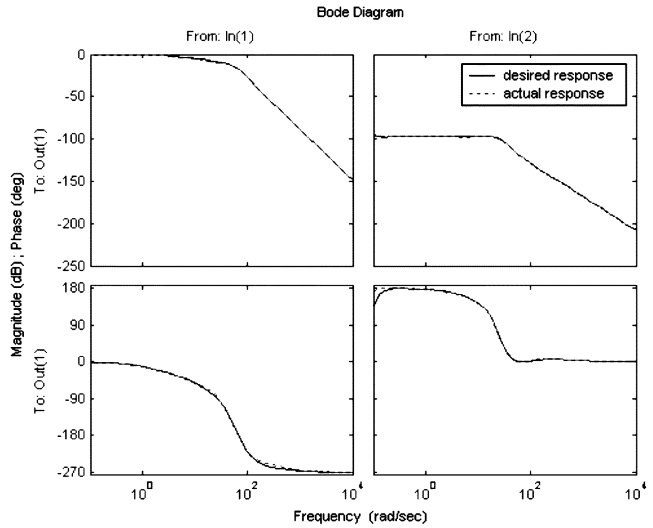


Fig. 10. Desired responses and actual responses of evolved suspension design. Left (road disturbance response). Right (load disturbance response).

An active suspension system has the ability to store, dissipate and to introduce energy to the system, with extra flexibility to achieve improved design performance. It should be noted that in this work, we have assumed that the sensor and the actuator have perfect dynamics. The suspension design will be considerably modified if such assumptions do not hold well.

IV. Application to MEMS Design

A. Problem Description

A microelectromechanical (MEM) bandpass filter is used here as the second application for experimenting with knowledge interaction in the evolutionary synthesis process. Due to its multidomain and intrinsically three-dimensional nature, MEMS design and analysis is very complicated and requires access to simulation tools with finite element analysis capability, such as Conventorware or ANSYS. It is preferable at the beginning of MEMS modeling and design to use a high-level system model to reduce the number of degrees of freedom to as few as possible (resembling in this way the top-down design methods that are very successful in VLSI design) [33].

A MEMS model needs to be capable of encompassing multiple energy domains, which makes it suitable for representation as a bond graph. In describing the macro-level behavior of the system, the bond graph representation is also compatible with 3-D numerical simulation, as long as a suitable lumping of components can be maintained. In this work, the first step in MEMS synthesis is to develop a strategy to automatically generate bond graph models to meet particular design specifications on the system level. Then on the second or lower level, other numerical optimization approaches [34], as well as evolutionary computation [35], may be used to synthesize customized components from a functionality specification. Fig. 11 shows a typically structured MEMS synthesis procedure.

B. Domain Knowledge Incorporation

Realizability of design topologies is regarded as the most important domain knowledge in MEMS design, due to the

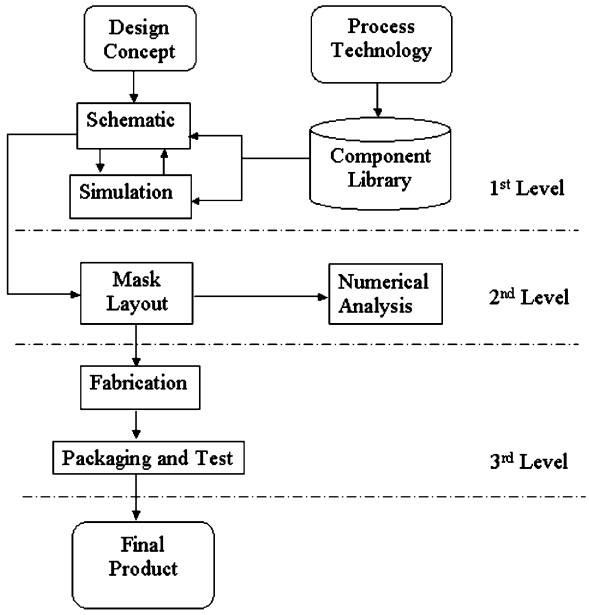


Fig. 11. Structured MEMS synthesis procedure.

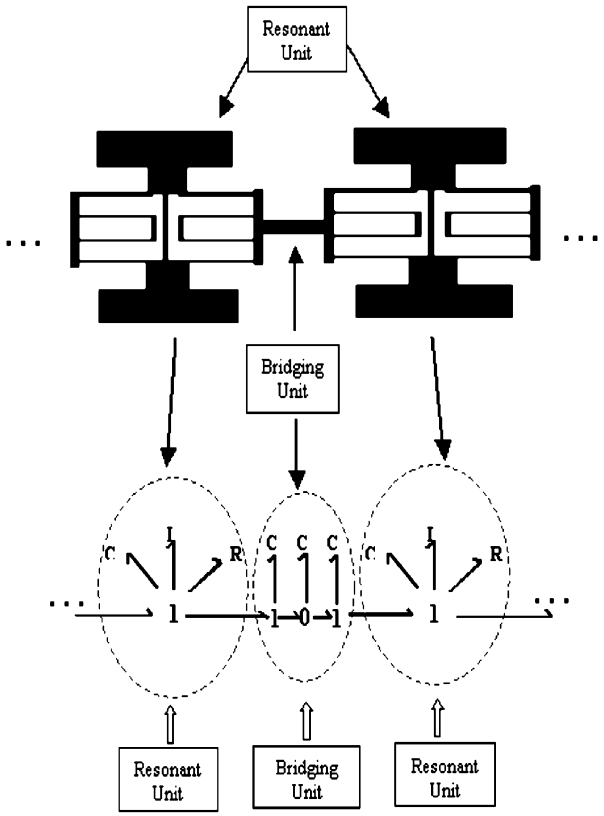


Fig. 12. Layout of filter topology I.

difficulty and complication of MEMS design and fabrication. It is advisable to adopt relatively mature and widely accepted topologies in the design process. Two popular topologies for a micromechanical bandpass filter, built using surface micromachining, are topologically composed of a series or concatenation of resonant units (RUs) and bridging units (BUs), or RUs and coupling units (CUs) [36], [37]. Figs. 12 and 13 illustrate the layouts and bond graph representations of

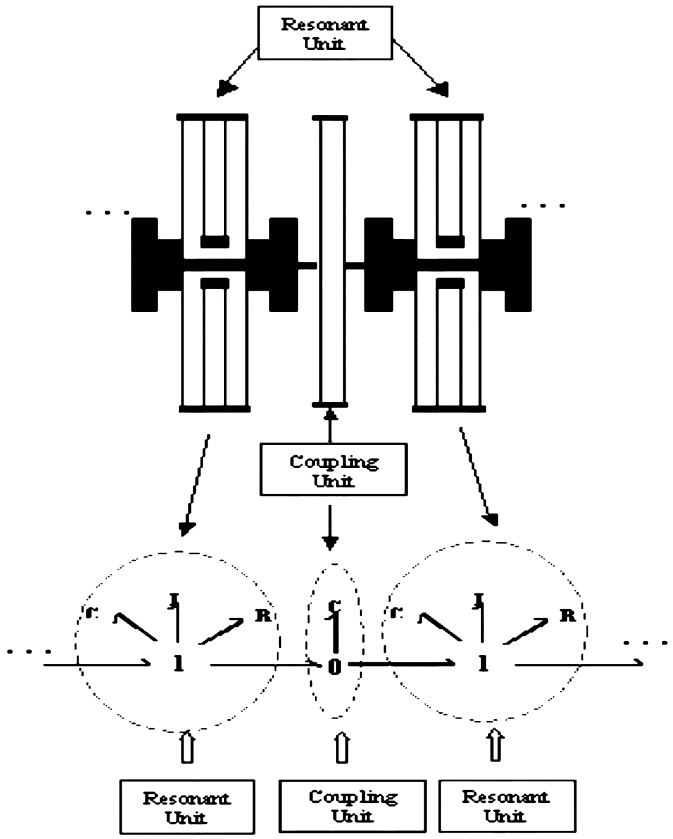


Fig. 13. Layout of filter topology II.

two such filter topologies, labeled I and II. The inclusion of realizable component topologies such as RUs and BUs in the MEMS filter complex module function set (see Table II) can impose domain knowledge of design constraints on the final synthesis results for guaranteed manufacturability of the design under current or anticipated manufacturing technology.

In Fig. 12, the filter is primarily composed of RUs and BUs, while in Fig. 13, the filter is primarily composed of RUs and CUs. For different design settings, alternative realizable function sets can be used to evolve different realizable structures for MEMS. The experiments reported in this paper used the following two realizable function sets:

$$\Re 1 = \{1-\text{junction}, C, R, I, +, -, \text{RU}, \text{CU}\} \quad (10)$$

$$\Re 2 = \{1-\text{junction}, C, R, I, +, -, \text{RU}, \text{BU}\}. \quad (11)$$

C. Experimentation and Results Analysis

The MEM filter design problem uses the bond graph model shown in Fig. 14 as the embryo. The accompanying block diagram indicates that the implementation will accept an electrical voltage signal as input and produce a voltage signal as output, but the interior components will be implemented as micromechanical elements. From a multiport network synthesis point of view, this is a two-port network, with specified immittance from output effort to input effort.

Filter performance is measured by the magnitude ratio of the frequency response for the voltage across R_L to the input voltage u_s. The desired frequency response has unity magnitude

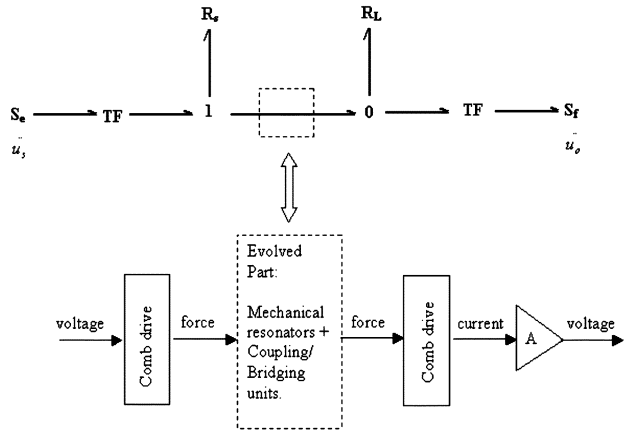


Fig. 14. MEM filter design embryo in bond graph and block diagram forms.

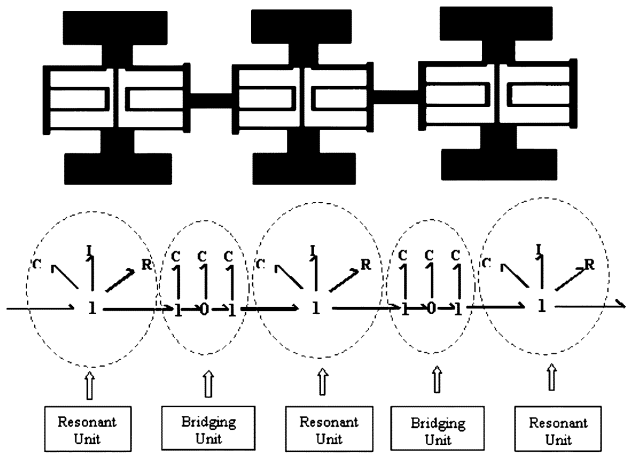


Fig. 15. Evolved MEM filter design candidate.

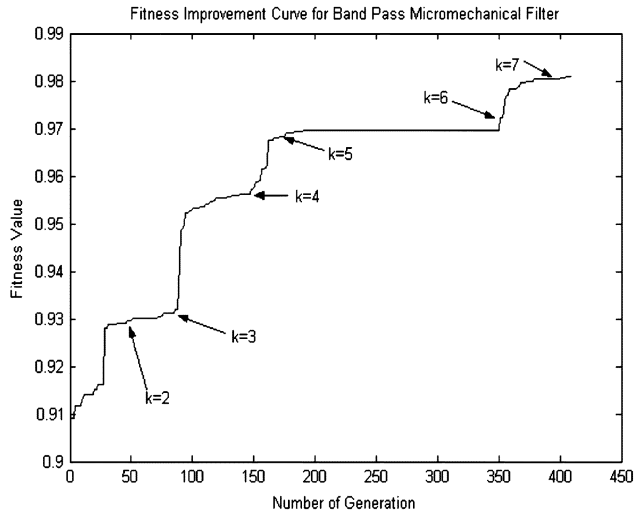


Fig. 16. Fitness improvement curve.

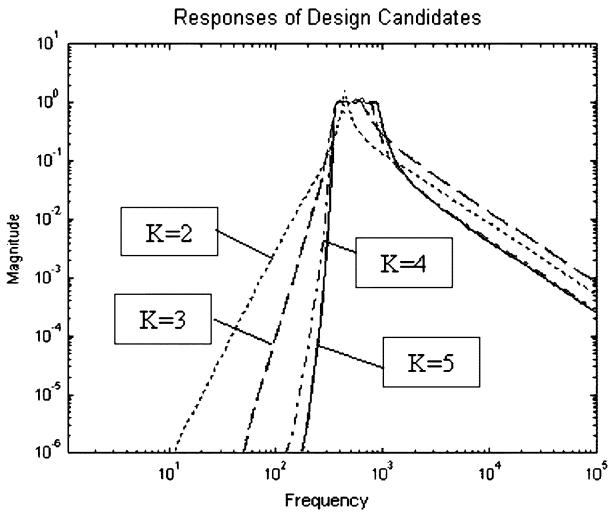


Fig. 17. Frequency responses of design candidates at different generations.

ratio in the pass band (316 Hz–1000 Hz), and zero magnitude ratio outside the pass band. The frequency range of interest is 0.1 Hz–100 kHz.

To evaluate fitness within the frequency range of interest, 100 points are sampled at equal intervals on a log scale. The magnitudes of the frequency response at the sample points are compared with their desired magnitudes. The differences are computed and the sum of all squared differences is taken as raw fitness, defined as $\mathrm{Fitness}_{\mathrm{raw}}$. The normalized fitness is calculated according to

$$\mathrm{Fitness}_{\mathrm{norm}} = 0.5 + \mathrm{norm}/(\mathrm{norm} + \mathrm{Fitness}_{\mathrm{raw}}) \qquad (12)$$

where *norm* is the scaling factor.

Results of the experiments show the capability of the proposed approach for finding realizable designs for micro-electromechanical filters. Layout of a design candidate and its bond graph representation evolved using realizable function set $\Re 2$ in (11) is shown in Fig. 15, with three resonators and two bridging units. Notice that the geometry of resonators may not show the real sizes and shapes of a physical resonator and the layout figure only serves as a topological illustration.

Fig. 16 shows the fitness improvement curve of a typical genetic programming run, in which K is defined as the number of resonator units used in the MEM filter design. It is shown that as evolution progresses, the fitness value undergoes continual improvement. It is also observed that as fitness improves, the value of K also becomes larger. This observation is supported by the reasoning that a higher order system with more resonator units has the potential of having better system performance than its lower order counterpart.

The plot of corresponding system frequency responses at generations 27, 52, 117, and 183 are shown in Fig. 17, with increased K value and performance evaluation.

The use of realizable function sets can be made less rigid to assist the designer in exploring more novel topologies for MEM filter design. The designer may use a function set in which not all elements are guaranteed to be strictly realizable. Instead, a different set of design knowledge is incorporated in the evolutionary process—i.e., a semirealizable function set may be used to relax the topological constraints with the purpose of finding new topologies not discovered before but still usually realizable after careful design. Fig. 18 gives an example of a novel topology evolved for a MEM filter design by incorporating a special CIR component (in Table II) into the semirealizable function set.

The work presented in this section analyzes the promise of MEMS design synthesis at the system level using the BG/GP

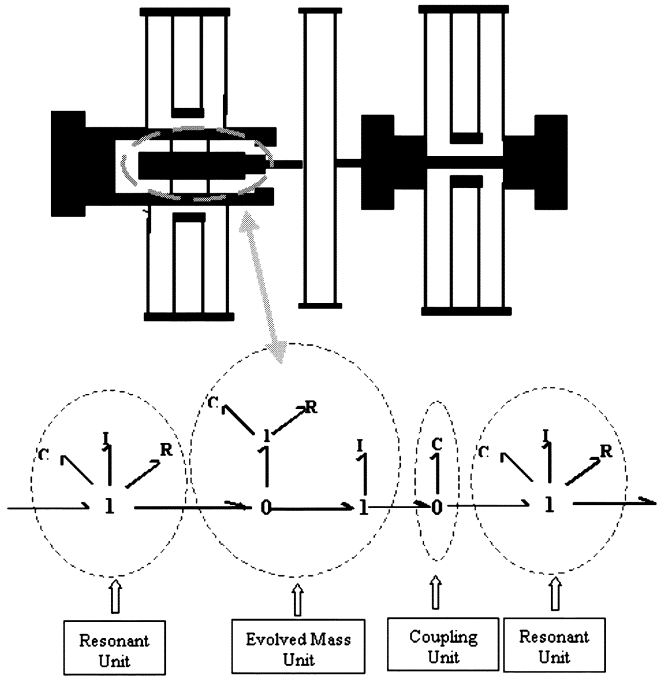


Fig. 18. Novel design topology using a semirealizable function set.

approach. The basic GP function set imposes very few constraints on design, while the realizable function set used for MEMS design features relatively few but structurally more complex devices in the component library. The use of a realizable function sets guarantees that the phenotypes generated can be built using existing or expected manufacturing technology. Along this path, large-scale component reuse and assembly of MEMS is expected to show more applicability and promise of this method for MEMS design.

V. Conclusion

This work extends the previous basic BG/GP approach to enhance human understanding of the mechatronics evolutionary synthesis process by concentrating on knowledge interaction with evolutionary computation. It provides a unified approach to evolutionary synthesis of mechatronic systems that facilitates significant knowledge-based interaction with evolutionary algorithms by structuring the design primitives into modular building blocks, both inductively and deductively.

Bond graphs are established to be very well suited to mechatronics system modeling and knowledge representation, because of their rich expressiveness across various engineering domains. The combination of bond graphs with genetic programming, with a fairly direct mapping from genotype to phenotype, provides a powerful capacity to explore topologically open-ended search spaces and extract useful domain knowledge. The use of realizable function sets reduces the GP search space by incorporating modular domain-specific knowledge into the design process to improve the efficiency of computational synthesis and the realizability of the designs generated. It also provides the designer with capability to interactively control the balance between design space exploration and exploitation by adjusting what is included in the realizable function set.

There are many potential research directions for future work. For example, this work integrates physical system modeling with control system design using the same design language—bond graphs, thus providing a concurrent engineering approach suitable for design of 'physical body' and 'active brain' simultaneously, which appears to be a promising direction for coevolutionary development of both systems to achieve overall design optimality. Also, the applications studied in this work require only linear system synthesis. It is also possible to include nonlinear effects for mechatronics design using bond graphs.

This work mainly addresses incorporating domain-specific knowledge into design practice. Design is multifaceted, it would also be desirable to incorporate other types of knowledge into evolutionary synthesis—for example, preference articulation [6]–[8], etc. It is our goal to continue working on developing a human-computer interactive design infrastructure to solve real-world engineering design problems, into which preference articulation, constraint handling, domain knowledge, and evolutionary computation can be integrated with great flexibility.

References

[1] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic," *J. Math. Comput. Sim.*, vol. 60, pp. 245–276, Sep. 2002.

[2] C. Wang, T. Hong, and S. Tseng, "Integrating fuzzy knowledge by genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 2, no. 4, pp. 138–149, Nov. 1998.

[3] Y. Jin and B. Sendhoff, "Knowledge incorporation into neural networks from fuzzy rules," *Neural Process. Lett.*, vol. 10, no. 3, pp. 231–242, 1999.

[4] J. J. Grefenstette, "Incorporating problem specific knowledge into genetic algorithms," in *Genetic Algorithms and Simulated Annealing*, L. D. Davis, Ed. San Mateo, CA: Morgan Kaufmann, 1987, pp. 42–60.

[5] S. J. Louis, "Genetic learning for combinational logic design," in *Proc. GECOO—Approximation and Learning in Evolutionary Computation Workshop*, 2002, pp. 21–26.

[6] Y. Jin and B. Sendhoff, "Fuzzy preference incorporation into evolutionary multi-objective optimization," in *Proc. 4th Asia-Pacific Conf. Simulated Evolution and Learning*, vol. 1, Singapore, Nov. 2002, pp. 26–30.

[7] D. Cvetkovic and I. C. Parmee, "Preferences and their application in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 42–57, Feb. 2002.

[8] J. Wang and J. P. Terpenny, "Interactive evolutionary solution synthesis in fuzzy set-based preliminary engineering design," *J. Intell. Manufact.*, vol. 14, no. 2, pp. 153–167, 2003.

[9] I. Pérez, C. A. C. Coello, and A. H. Aguirre, "Extracting and re-using design patterns from genetic algorithms using case-based reasoning," in *Genetic and Evolutionary Computation (GECCO) Workshop Program*, 2002, pp. 27–30.

[10] H. Takagi, "Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation," *Proc. IEEE*, vol. 89, no. 9, pp. 1275–1296, Sep. 2001.

[11] J. J. McPhee, "On the use of linear graph theory in multibody system dynamics," *Nonlin. Dynam.*, vol. 9, pp. 73–90, 1996.

[12] P. J. Angeline, "Genetic programming and emergent intelligence," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. Cambridge, MA: MIT Press, 1994.

[13] "Computational synthesis: From basic building blocks to high level functionality," in *AAAI Symp.*, H. Lipson, E. K. Antonsson, and J. R. Koza, Eds., Stanford, CA, Mar. 2003.

[14] E. D. Goodman, K. Seo, R. C. Rosenberg, Z. Fan, J. Hu, and B. Zhang, "Automated design methodology for mechatronic systems using bond graphs and genetic programming," in *2002 NSF Design, Service and Manufacturing Grantees and Research Conf.* , San Juan, PR, Jan. 2002, pp. 206–221.

[15] Z. Fan, K. Seo, J. Hu, R. C. Rosenberg, and E. D. Goodman, "System-level synthesis of MEMS via genetic programming and bond graphs," in *Proc. Genetic and Evolutionary Computing Conf.*. Chicago, IL, Jul. 2003, pp. 2058–2071.

[16] J. Wang and J. P. Terpenny, "Integrated active and passive mechatronic system design using bond graphs and genetic programming," in *Late Breaking Paper, Genetic and Evolutionary Computation (GECCO) Conf.*, Chicago, IL, Jul. 2003, pp. 322–329.

[17] N. Kyura and H. Oho, "Mechatronics—An industrial perspective," *IEEE/ASME Trans. Mechatronics*, vol. 1, no. 1, pp. 10–15, Mar. 1996.
[18] *The Mechatronics Handbook*, R. H. Bishop, Ed., CRC Press, Boca Raton, FL, 2003. R. Isermann, "Mechatronic Design Approach".
[19] W. W. Harman and D. W. Lytle, *Electrical and Mechanical Networks*. New York: McGraw-Hill, 1962.
[20] R. W. Newcomb, *Linear Multiport Synthesis*. New York: McGraw-Hill, 1966.
[21] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System Dynamics: A Unified Approach*, 3rd ed. New York: Wiley, 2000.
[22] J. F. Broenink, Introduction to Physical Systems Modeling with Bond Graphs, May 20, 1999. [Online]. Available:.
[23] S. Finger and J. R. Rinderle, "A transformational approach to mechanical design using a bond graph grammar," in *ASME 1989 Design Theory and Methodology Conf.*, Montreal, QC, Canada, 1989.
[24] K. Ulrich and W. Seering, "Synthesis of schematic descriptions in mechanical design," *Res. Eng. Design*, vol. 1, pp. 3–18, 1989.
[25] B. H. Wilson and B. Eryilmaz, "Using bond graphs to synthesize tree-structured transfer functions: Improved frequency-domain analysis of uncertain systems," *J. Franklin Inst.*, vol. 335B, no. 8, pp. 1443–1465, 1998.
[26] A. Sharon, N. Hogan, and 3-184(to)-23vpp75282ph-16r6(t91 T-16rD[)-326.3(Intro-292.4.3(.) 0 TD[(1465(pro)15,F4 91.189ankl9.3824ical)-258.7(design,)]TJ/46 1 Tf