GPBG: A Framework for Evolutionary Synthesis of Multi-domain Engineering Systems

Jianjun Hu[1], Zhun Fan[4], Jiachuan Wang[5], Kisung Seo[7], Xiangdong Peng[2], Janis Terpenny[6], Ronald Rosenberg[3], and Erik Goodman[2]

[1] MCB 403G, University of Southern California, Los Angeles, CA, 90089, USA. jianjunh@usc.edu
[2] Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824, USA. goodman@egr.msu.edu
[3] Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824 rosenber@egr.msu.edu
[4] Department of Mechanical Engineering, Technical University of Denmark, Building 404, DK-2800 Lyngby, DENMARK. zf@mek.dtu.dk
[5] United Technologies Research Center, Systems Department, East Hartford,CT 06108, USA. wangj2@utrc.utc.com
[6] Department of Engineering Education, Virginia Tech, Blacksburg, VA 24061, USA. terpenny@vt.edu
[7] Department of Electronics Engineering, Seokyeong University, Seoul, 136-704, KOREA. ksseo@skuniv.ac.kr

Summary. This chapter presents a generic framework (GPBG) for evolutionary design synthesis of multi-domain engineering systems by exploiting the open-ended topological search capability of genetic programming and the multi-domain modeling of bond graphs. In this framework, an engineering design problem is solved in four steps, namely, design space definition, functional specification, evolutionary search, and design implementation. We used four design problems to demonstrate the effectiveness of this methodology, including synthesis of vibration absorbers, synthesis of MEMS filters, and synthesis of suspension systems to illustrate the basic principles of GPBG, the means to incorporate domain-knowledge into the evolutionary design process, and the potential of GPBG for design innovation.

1 Introduction

Current engineering design is a multi-step process proceeding from conceptual design to detailed design and to evaluation and testing. It is estimated that 60-70% of design decisions and most innovation occur in the conceptual design stage, which may include conceptual design of function, operating principles, layout, shape, and structure. However, few computational tools are available

to help designers to explore the design space and stimulate the product innovation process. As a result, product innovation is strongly constrained by the designer's ingenuity and experience, and a systmatic approach to product innovation is strongly needed.

Many engineering design problems, such as mechatronic systems, can be abstracted as a design space exploration problems in which a set of building blocks or modules need to be assembled/connected together to compose a system satisfying a set of given functional requirements. In many cases, such as analog circuit design, it is relatively easy to simulate a product design model to evaluate its functional performance via simulation software such as P-SPICE, while it is extremely hard to come up with an innovative design solution out of the almost unlimited number of design candidates of the topological design space. An efficient topological search technique is needed to help to improve this process.

In recent decades, evolutionary computation has emerged as an effective and promising search/optimization technique that is suitable for large-scale non-linear multi-modal engineering optimization problems. In particular, genetic programming (GP) has been used as an attractive approach for engineering design innovation in a variety of domains, including design of analog circuits, digital circuits, chemical molecules, control systems, etc. [11]. Such work employs GP as a topologically open-ended search technique for functional design innovation – achieving given behaviors without pre-specifying the design topology – and has achieved considerable success. While electrical circuits and block diagrams are well suited for the design problems in analog circuit design and controller synthesis, many engineering design problems cover multiple domains, including, for example, mechanical, electrical, and hydraulic subsystems. Since 2001, we have been developing a new framework, called GPBG [16], for automated synthesis of multi-domain systems using genetic programming and bond graphs, which are a well-established modeling tool for multi-domain systems.

In this chapter, we will detail how an engineering design problem can be solved under the GPBG framework using several design synthesis problems: a vibration absorber, a MEMS filter design, and a controller design in suspension systems. The rest of the chapter is organized as follows. Section 2 presents a survey of applications of evolutionary algorithms in engineering design synthesis that are more than just parameter optimization. Section 3 introduces the GPBG framework, which exploits genetic programming and bond graphs for automated synthesis of dynamic systems. In section 4, first, the vibration absorber design problem is used to illustrate the basic approach to mapping the engineering design problem into a topology space search problem using genetic programming. We then use a MEMS filter design problem to show how expert domain knowledge can be incorporated into the evolutionary synthesis and greatly improve the efficiency of this approach. The third application uses a different, direct-encoding GPBG approach for synthesizing

controllers of suspension systems. Finally, the conclusions and future research are highlighted in Section 5.

2 Related Work

Automated synthesis of dynamic systems has been investigated intensively in the past ten years. Most of that work is related to analog circuit synthesis, as pioneered by Koza and his colleagues[9][11]. Their work in automated analog circuit synthesis, including low-pass, high-pass, and asymmetric band-pass filters, is described in [10] [9]. Lohn and Colombano [13] proposed a linear representation approach to evolve analog circuits. Ando and Iba [1] suggested another simple linear genome method to evolve low-pass and band-pass filters with small numbers (<50) of components. In our previous work, we applied GP to the lowpass analog filter design problem[2], MEMS [3], the printer mechanism design, active-passive dynamic system design[17], all using bond graphs as the modeling and simulation tool. Controllers, or dynamic systems represented as block diagrams have also been synthesized automatically using genetic programming by Koza et al. [12]. This work has led to the invention of a patentable controller having better performance than a standard PID controller.

3 The GPBG Framework for Evolutionary Design

In this section, we present a generic methodology for open-ended computational synthesis of multi-domain dynamic systems based on bond graphs [8] and genetic programming–the GPBG approach = Genetic Programming+Bond Graphs.

3.1 Genetic Programming

Genetic programming is a derivative of genetic algorithms that is characterized by its capability to evolve programs. In typical GP, an individual is represented as a syntax/GP tree composed of functions and terminals defined by the user according to the problem. Each function has one or more inputs, while terminals have no inputs. Both functions and terminals can be executed to generate some output or do some processing such as inserting a new component into a developing/growing analog circuit. Genetic programmingś open-ended topological search capability has been widely applied to computational synthesis of analog circuits, controllers, mechatronic systems, quantum circuits, etc.

3.2 Bond Graphs

The bond graph is a multi-domain modeling tool for analysis and design of dynamic systems, especially hybrid multi-domain systems, including mechanical, electrical, pneumatic, hydraulic, etc., components. Details of notation and methods of systems analysis related to the bond graph representation can be found in [8]. Fig. 1 illustrates a small bond graph that represents the accompanying electrical system.

A typical simple bond graph model is composed of inductors (I), resistors (R), capacitors (C), transformers (TF), gyrators (GY), 0-Junctions (J0), 1-junctions (J1), sources of effort (SE), and sources of flow (SF). In this chapter, we are only concerned with linear dynamic systems and do not include transformers and gyrators as components.

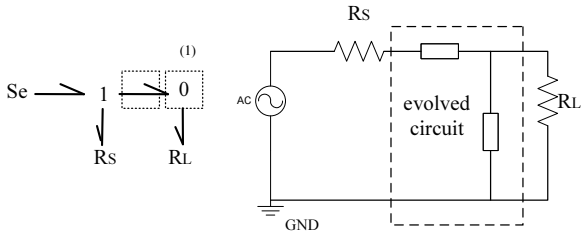

Fig. 1. A bond graph and its equivalent circuit. The dotted boxes in the left bond graph indicate modifiable sites at which further topological manipulations can be applied (to be explained in the next section)

In the context of electric circuit design, a bond graph consists of the following types of elements:

- C, I, and R elements, which are passive one-port elements that contain no sources of power, and represent capacitors, inductors, and resistors
- Power source elements including Se and Sf, which are active one-port elements representing sources of voltage and current, respectively. In addition, when the current of a current source is fixed as zero, it can serve as an ideal voltage gauge. Similarly, when the voltage of a voltage source is fixed as zero, it can serve as an ideal current gauge
- Transformer (TF) and gyrator (GY), which are two-port elements. Power (i.e., product of voltage and current) is conserved in these elements, but the values of voltage and current may be changed in the elements
- 0-junctions and 1-junctions, which are multi-port elements for representing series and parallel relationships among elements. They serve to interconnect elements into subsystem or system models
- Bonds, which are used to connect any two elements in the bond graph

A unique characteristic of bond graphs is their use of 0- and 1-junctions to represent the series and parallel relationships among components in circuits. In fact, it is this concept that led to the foundation of the bond graph field [14]. Junctions transform common circuits into a very clean structure with few loops, which can otherwise make circuits appear very complicated. Figure 4.1 shows the comparison of a circuit diagram and a corresponding bond graph. The evaluation efficiency of the bond graph model is further improved due to the fact that analysis of causal relationships and power flow between elements and subsystems can reveal certain system properties and inherent characteristics. This makes it possible to discard infeasible design candidates even before numerically evaluating them, thus reducing time of evaluation to a large degree. In addition, as virtually all of the circuit topologies created are valid, our system does not need to check validity conditions of individual circuits to avoid singular situations that could interrupt the running of a program evaluating them.

3.3 GPBG = GP + Bond Graphs

By combining the topological search capability of GP and the multi-domain representation feature of bond graphs, GPBG provides an appealing approach for open-ended synthesis of multi-domain systems. To map an engineering design problem into the GPBG framework, the design space of target design solutions must first be identified, including all component types, component interfaces, and connection types. Then, depending on how we encode the bond graphs using the GP tree, two types of approaches have been used within GPBG framework. One is the developmental GPBG approach, similar to Koza's work on evolving analog filters, in which the bond graph phenotypes are grown from an embryo bond graph by executing the GP tree program to manipulate the topology. The other approach is the direct encoding GPBG, in which the the GP trees directly encode the bond graph topology. This method shares some similarity to the GP-based controller synthesis approach by Koza [12].

Developmental GPBG

The problem of automated synthesis of bond graphs involves two basic searches–the search for a good topology and the search for good parameters for each topology–in order to be able to evaluate its performance. Based on Koza's work [9] on automated synthesis of electronic circuits, we created a developmental GPBG system for synthesizing mechatronic systems, including:

1. An embryo bond graph with modifiable sites at which further topological operations can be applied to grow the embryo into a functional system

2. A GP function set, composed of a set of topology manipulation and other primitive instructions which will be assembled into a GP tree by the evolutionary process (execution of this GP program leads to topological and parametric manipulation of the developing embryo bond graph)
3. A fitness function to evaluate the performance of candidate solutions

We use the analog filter synthesis problem as an example to illustrate the developmental GPBG approach[2]. In this problem, the design space consists of bond graphs composed of capacitors (C), inductors (I), resistors (R), 1-junctions (J1) and 0-junctions (J0) (we omitted transformers and gyrators for the sake of simplicity). We have two types of elements in a bond graph. One is the elements including C/I/R/J1/J0 which have one or more interface ports. The second type of elements are two-port bonds. The applicable topological operations on node elements include replacing the component type (on C/I/R/J1/J0), and adding a new C/I/R component (on J1/J0). The topological operations on bonds include inserting a new J1/J0. This operator set will enable the GPBG to evolve a large number of bond graphs. Then we develop an alternative, "basic" GP function set for this problem, including {Insert_J0/J1, Add_C/I/R, and Replace_C/I/R}. Fig.2 and Fig.3 shows how these topological manipulation GP functions work. Note that for Add_C/I/R functions, we can have one or more branches that accept numeric subtrees to set the component parameters. More examples of developmental GP are presented in Section 4.1 and Section 4.2.

The developmental GPBG framework enables us to do simultaneous topology and parameter search. Compared to the direct encoding GPBG below, it can evolve much more diverse topology types including those with loops and has more flexibility during the evolutionary search process, but at the cost of a less intuitive GP tree.

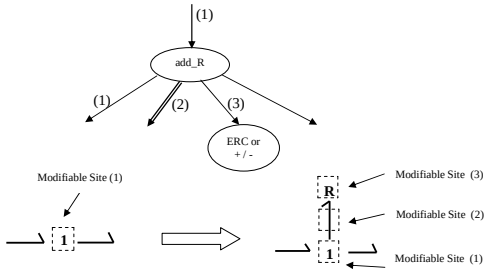

Fig. 2. The add_R function adds a resistor to a junction.

GPBG with direct encoding

One interesting observation of typical bond graphs is that most bond graphs do not have loops and are themselves tree-structured. This makes it natural

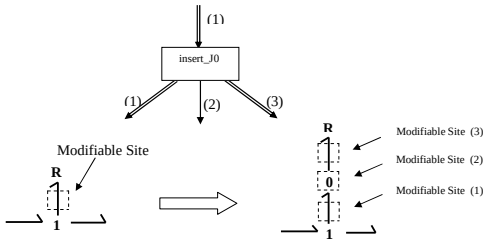

Fig. 3. The Insert_J0 inserts a new 0-junction into a bond

to use the GP trees themselves to represent/encode the bond graph structure. We thus proposed the direct-encoding GPBG approach for evolving tree-type bond graph models. In this approach, 1-junction and 0-junction are used as GP functions with two input variables (ports). The capacitors/resistors/inductors are all GP functions with one input that connects to a numeric subtree to establish the component size (parameter value). We also have plus/minus arithmetic operators in addition to ERC random terminals. One such function set is shown in Table 1, which is used to synthesize controllers for a suspension system in Section 4.3. Since there is a one-to-one correspondence of the GP tree and bond graph topology, one can easily build a bond graph by following the topologies in the GP trees.

Table 1. GP function set for suspension controller synthesis

Name and description	Function arity
J0 - Junction (0)	2
J1 - Junction (1)	2
R Element (R)	1
C Element (C)	1
I Element (I)	1
Arithmetic + (+): add two ERCs	2
Arithmetic - (-): Subtract two ERCs	2
Ephemeral Random Constant (ERC)	0

With this encoding approach, each GP tree is a bond graph. Actually, here, all GP trees are binary trees. Clearly, this can only represent a subset of bond graphs compared to the developmental GPBG approach, but enjoys simplicity in implementation. Since the standard GP needs to specify the arity of the GP functions, this limits the number of ports of the junctions to be three, which is a shortcoming of this approach. However, this disadvantage can be ameliorated by defining a larger arity, (e.g., 8) and then definoing a null-element terminal. In this way, we can evolve 6-arity GP trees in which many of the ports are simply empty. Most real-world bond graph models have

fewer than 8 ports. One possible disadvantage maybe that this may greatly increase the search space.

4 Case Studies of Evolutionary Synthesis Using GPBG

In this section, we applied GPBG to four real-world design problems, including synthesis of a passive vibration absorber, a MEMS filter, a robust analog filter, and a controller for a suspension system. The purpose of the first example is to illustrate the basic principle of GPBG with practical design problems. The second example aims to show how a designer can combine domain knowledge with the evolutionary design process. The third example is to show how the evolutionary design based on GPBG can provide solutions to unconventional design problems such as design for robustness. Each problem will start with the description of the design space and the configuration of GPBG, including design embryo, GP function set, and fitness function.

4.1 Synthesis of Mechanical Vibration Absorber

Problem Description

Vibration absorbers are a class of dynamic systems, and can be modeled as analog circuits, block diagrams, bond graphs, etc. One special characteristic of these particular dynamic systems is that the building blocks usually have a fixed number interface ports and may not be connected arbitrarily.

In this section, we are mainly interested in synthesizing passive vibration absorbers to reduce the vibration response of primary systems of various configurations. Figure 4 shows a primary system and its corresponding bond graph model. The design task is to attach some new components to the primary system such that the frequency response at the excitation frequency ω be minimized. Figure 5 shows the first vibration absorber, invented by H. Frahm in 1911, and its bond graph model. The frequency response of the stand-alone primary system and the primary system with vibration absorber is shown in Figure 6. It can be seen that the vibration absorber can significant quench the response of the primary systems at the excitation frequency.

In this design problem, the objective is to synthesize a vibration absorber such that the frequency response

$$f_{raw} = |TF(j\omega)|_{\omega=\omega_0} \tag{1}$$

of the primary system mass (displacement) at the frequency ω of excitation force $f = f_0 * sin\omega t$ is minimized. This problem is extracted from [7]. We want to see if the GPBG system can reinvent the first patented vibration absorber, shown in Figure 5. The parameters of the primary system are as follows:

m_p = 5.77 kg; k_p=251.132 *1e6 N/m; c_p= 192.92 kg/s.

The parameters of the standard passive absorber solution are as follows:

m_a = 0.227 kg; k_a=9.81e6 N/m; c_a= 355.6 kg/s

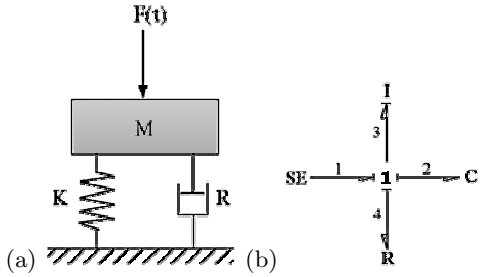

Fig. 4. The bond graph structure of a primary system and its bond graph model (a) The primary system under perturbation of excitation force F(t); (b)The bond graph model of the embryo system.

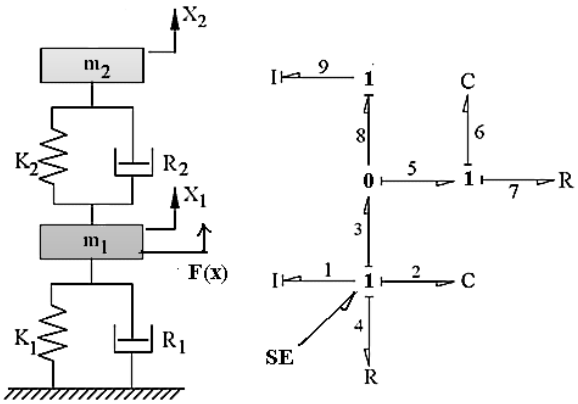

Fig. 5. The bond graph structure of the first patented vibration absorber and its bond graph model.

GPBG Configuration

The design space of passive vibration absorbers is composed of masses (R), springs (C), and dampers, corresponding to Resistor(R), Capacitor (C) and Inductor (I), respectively. Following the GPBG framework outlined before, we used the bond graph embryo in Figure1 for this design problem. The modifiable site is the 1-junction. Since it is not physically realistic to have many masses attached to the primary structures, we limit the maximum number of masses to 2 in all the experiments.

In our earliest work [2], a "basic" GP function set was used for evolutionary synthesis of analog filters. In that approach, the GP functions for topological operation included {Insert_J0/J1, Add_C/I/R, and Replace_C/I/R}, which allowed evolution of a large variety of bond graph topologies. The shortcoming of this approach is that it tended to evolve redundant and sometimes causally ill-posed bond graphs [15]. Later, we used a causally well-posed modular GP

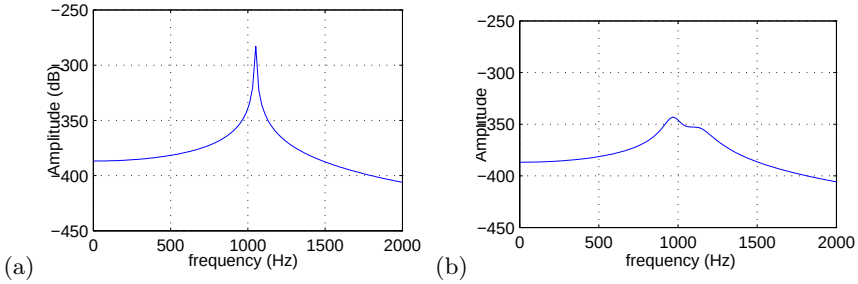

Fig. 6. Frequency responses of the primary system under perturbation of excitation force F(t), without and with a vibration absorber. (a) without vibration absorber (b) with vibration absorber.

function set to evolve more concise bond graphs with much less redundancy [6]. However, that encoding had a strong bias toward a chain-type topology and thus may have limited the scope of topology search [5]. Here we have improved the basic function set in [2] and developed the following hybrid function set approach to reduce redundancy while enjoying the flexibility of topological exploration:

`F={ Insert_J0E, Insert_J1E, Add_C/I/R, EndNode, EndBond, ERC}`

where the Insert_J0E, Insert_J1E functions insert a new 0/1-junction into a bond while attaching at least one and at most three elements (from among C/I/R). EndNode and EndBond terminate the development (further topology manipulation) at junction modifiable sites and bond modifiable sites, respectively; ERC represents a real number (Ephemeral Random Constant) that can be changed by Gaussian mutation. In addition, the number and type of elements attached to the inserted junctions are controlled by three "flag" bits. A flag mutation operator is used to evolve these flag bits, each representing the presence or absence of the corresponding C/I/R component. Compared with the basic function set approach, this hybrid approach can effectively avoid adding many bare (and redundant) junctions. At the same time, Add_C/I/R still provides the flexibility needed for broad topology search. For any of the three C/I/R components attached to each junction, there is a corresponding parameter to represent the component's value, which is evolved by a Gaussian mutation operator in the modified genetic programming system used here. This is different from our previous work in which the "classical" numeric subtree approach was used to evolve parameters of components. Fig. 9 shows a GP tree that develops an embryo bond graph into a complete bond graph solution. Our comparison experiments [5] showed that this function set was more effective on both an eigenvalue and an analog filter test problem.

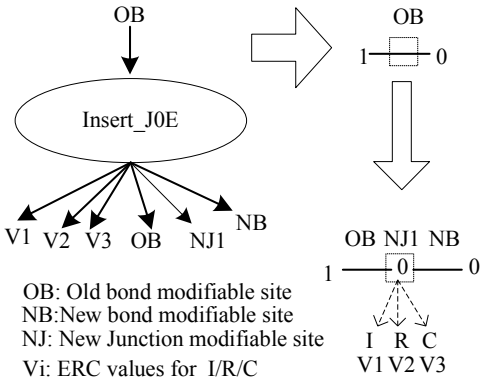

Fig. 7. The Insert_J0E GP function inserts a new junction into a bond along with a certain number of attached components

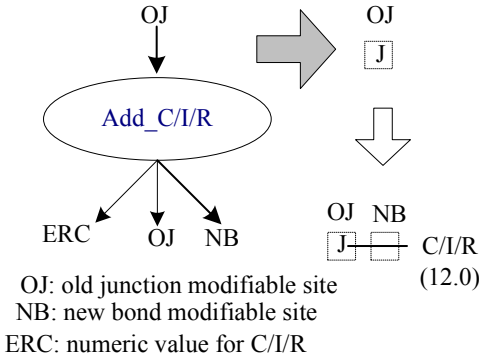

Fig. 8. The Add_C/I/R GP function adds a C/I/R component to a junction

The fitness function for candidate design evaluation is defined as:

$$f_{norm} = \frac{NORM}{NORM + f_{raw}} \tag{2}$$

where f_{raw} is the frequency response as defined in Equation 1. NORM is a normalization term aimed at adjusting the f_{norm} into the range of [0,1]. This process transforms the minimization of deviation from target frequency response into a maximization of fitness process as used in our GP system. Since tournament selection is used as the selection operator, the normalization term can be an arbitrary positive number. Here NORM is set to 10, and the fitness range is [0, 1].

According to Eq.1, we need to calculate the frequency response $\frac{X_1(s)}{F(s)}$ where X_1 is the displacement of the primary mass. However, we can only extract from a bond graph the source effort signal $\dot{X}\ (s)$. We use the following procedure to get the f_{raw}:

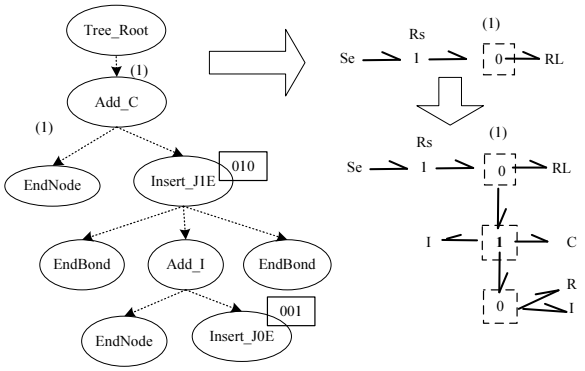

Fig. 9. An example of a GP tree, composed of topology operators applied to an embryo, generating a bond graph after depth-first execution (numeric ERC nodes are omitted). Note that the 010 and 001 are the flag bit sets showing the presence or absence of attached C/I/R components

1)calculate A, B, C, D matrices from a given bond graph;
2)convert A, B, C, D into transfer function TF_{raw};
3)$TF_{norm} = TF_{raw} * 1/s$ is equal to $\frac{X_1(s)}{F(s)}$;
4)convert TF_{norm} back to A', B', C', D' matrices and simulate its frequency response with Matlab.

Design Experiments

Compared to the evolutionary synthesis of electrical circuits, a mechanical vibration absorber usually has a much smaller number of components. So the topological and parameter search can be greatly decreased. We used a bond graph simulation engine and developed the GPBG platform based on Open beagle GP framework by Christian Gagne[4]. Most of the experiments are finished in less than an hour. Some of them take only a few minutes. Here we set the maximum number of components to be 7. Other standard GP parameters are summarized in Table 4.1.

Figure 10 shows an evolved single frequency vibration absorber and its frequency response compared to the responses of the primary structure without any absorber and with a standard passive absorber invented in 1912. It is very interesting that the frequency response of the evolved vibration absorber has a very deep spike at the excitation frequency to minimize the frequency response at that single frequency. If the excitation frequency is relatively constant with little shifting, our evolved absorber will achieve better performance at that specific frequency. Our evolved vibration absorber utilizes one damper (I) and several springs (C), sharing similarity to the original absorber invention of 1912. We found the GPBG framework is very flexible for vibration absorber synthesis. In addition to this single-frequency vibration absorber, we

Table 2. Experimental parameters for vibration absorber synthesis

Parameter	Value	Parameter	Value
No. of subpopulations	5	Tournament Selection Size	7
Sub population size	400	pCrossover	0.4
Maximum evaluation	100000	pMutationStandard	0.05
Migration Interval	5 gen	MutateMaxDepth	3
Migration Size	40	pMutationParameter	0.3
Init.MaxDepth	3	pSwitchBit	0.2
Init.MinDepth	2	pSwapSubtree	0.05
StronglyTyped	True	TreeMaxDepth	7

have also synthesized novel dual frequency and band-pass passive vibrators, which will be reported elsewhere.

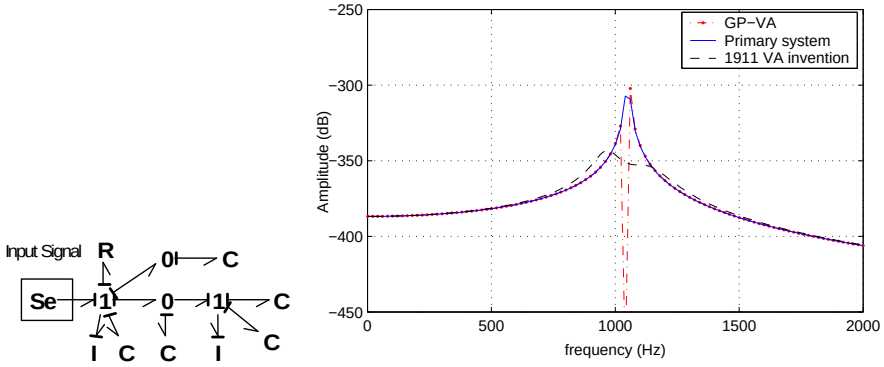

Fig. 10. The evolved single-frequency vibration absorber and its performance compared to a standard vibration absorber.

4.2 Synthesis of MEMS Filters: Knowledge Incorporation in GPBG

Due to the complexity of real-world engineering design problems, hands-free automated synthesis system can rarely provide entirely satisfactory solutions. It may be that the computational demand is too high for current inexpensive computing hardware or the design solutions are hard to implement using physical components or they violate some design constraints. It is thus strongly desirable to incorporate expert/human knowledge into the evolutionary synthesis process to create some kind of interactive evolutionary synthesis tools that help human designers make better decisions and explore under-explored design spaces.

Problem Description

In this section, we try to synthesize MEMS (micro-electro-mechanical systems) band-pass filters to examine how domain knowledge can be conveniently included into the evolutionary synthesis process. Due to its multi-domain and intrinsically three-dimensional nature, MEMS design and analysis is very complicated. However, the multi-domain property of MEMS models makes them suitable for representation as bond graphs. In this MEMS filter synthesis problem, the goal is to automatically generate bond graph models of MEMS filters to meet particular design specifications.

One distinct characteristic of MEMS filter design with vibration absorber synthesis and analog filter design is that, due to manufacturing constraints, MEMS filters are usually composed of a restricted and specialized set of components. Two popular topologies for micromechanical band-pass filters, 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]. Fig. 11 illustrates the layouts and corresponding bond graph representations of two such filter topologies, labeled I and II.

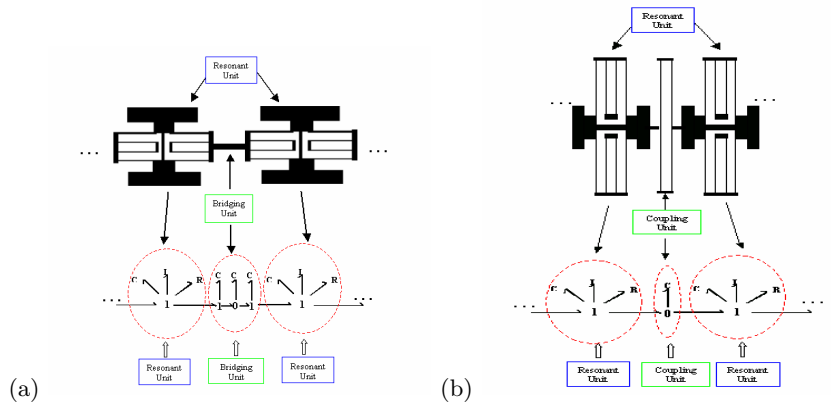

Fig. 11. MEMS filter topologies. (a) Layout of filter topology I (b) Layout of filter topology II.

From this figure, it is clear that here the building blocks of MEMS filter design are high-level modules that are tailored to a specific fabrication process. Design solutions composed of arbitrary topologies of basic primitive components will be difficult to manufacture.

GPBG Configuration

For this band-pass filter design problem, we use the bond graph model shown in Fig.12 as the design embryo of the GPBG framework. The accompanying block diagram indicates that this 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.

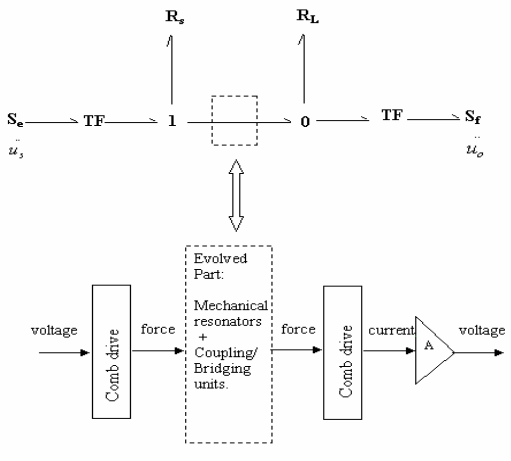

Fig. 12. MEM filter design embryo in bond graph and block diagram forms.

To incorporate the domain knowledge of MEMS filters, we propose the realizable GP function set concept, which manipulates topologies composed of manufacturable modules by adding, removing, or replacing these modules as units. We use the following GP function set in Table 4.2 as our modular GP function set, which can impose domain knowledge of design constraints on the final synthesis results for guaranteed manufacturability of the design under current or anticipated manufacturing technology. By using only operators in a realizable function set, we seek to guarantee that the evolved design is physically realizable and has the potential to be manufactured. This concept of realizability may include stringent fabrication constraints to be fulfilled in some specific application domain.

Examples of operators, namely insert_CU and insert_RU, are illustrated in Fig. 13. Examples of basic operators are available in our earlier work [2]. Figure 5.11 explains how the insert_BU function works. A Bridging Unit (BU) is a subsystem composed of three capacitors with the same parameters, attached together with a 0-junction in the center and 1-junctions at the left and right ends. After execution of the insert_BU function, an additional modifiable site (2) appears at the rightmost newly created bond. As illustrated in Figure 5.12,

Table 3. Realizable Function Set for MEMS Filter Synthesis

insert_RU	Insert a Resonator Unit
insert_CU	Insert a Coupling Unit
insert_BU	Insert a Bridging Unit
add_RU	Add a Resonator Unit
insert_J01	Insert a 0-1-junction compound with elements
insert_CIR	Insert a special CIR compound
insert_CR	Insert a special CR compound
Add_J	Add a junction compound
+	Sum two ERCs
-	Subtract two ERCs
endn	End terminal for add functions
endb	End terminal for insert functions
endr	End terminal for replace functions
erc	Ephemeral Random Constant (ERC)

a resonator unit (RU), composed of one I, R, and C component all attached to a 1-junction, is inserted in an original bond with a modifiable site through the insert_RU function. After the insert_RU function is executed, a new RU is created and one additional modifiable site, namely bond (3), appears in the resulting phenotype bond graph, along with the original modifiable site bond (1). The newly-added 1-junction also has an additional modifiable site (2). As components C, I, and R all have parameters to be evolved, the insert_RU function has three corresponding ERC-typed sites, (4), (5), and (6), for numerical evolution of parameters.

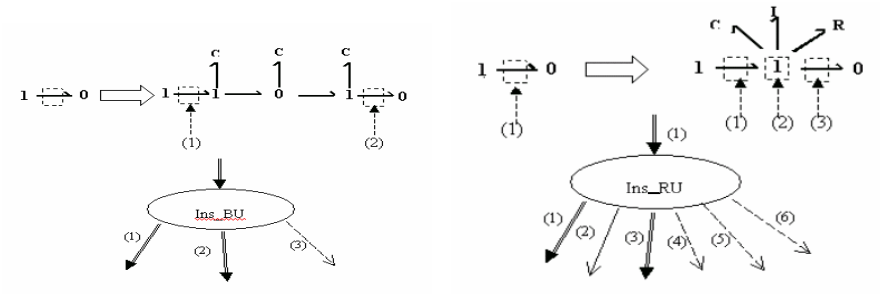

Fig. 13. Two realizable GP functions for MEMS filter design. (left) Insert_BU (right) Insert_RU

Filter performance is measured by the magnitude ratio of the frequency response for the voltage across RL to the input voltage u_s. The desired frequency response has unity magnitude ratio in the pass band (316Hz - 1000Hz), and zero magnitude ratio outside the pass band. The frequency range of interest

is 0.1Hz - 100KHz. 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. The normalized fitness is calculated according to Equation 2.

Design Experiments

We used a strongly-typed version of lilgp to generate bond graph models. The major GP parameters were as shown below.

Table 4. Experimental parameters for vibration absorber synthesis

Population size	500 in each of 13 subpopulations
Initial population	half_and_half
Initial depth	4-6
Max depth: 50	Max_nodes 5000
Selection	Tournament (size=7)
Crossover: 0.9	Mutation: 0.3

Results of the experiments show the capability of the GPBG approach for finding realizable designs for micro-electro-mechanical filters. Fig. 14(a) 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 system frequency responses at generations 27, 52, 117 and 183 are shown in Fig.14(b), 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 MEMS 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 semi-realizable 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 interpretation. Fig.15 gives an example of a novel topology evolved for a MEM filter design by incorporating a special CIR component (in Table II) into the semi-realizable function set.

The work presented in this section analyzes the promise of MEMS design synthesis at the system level using the GPBG approach. The basic GP

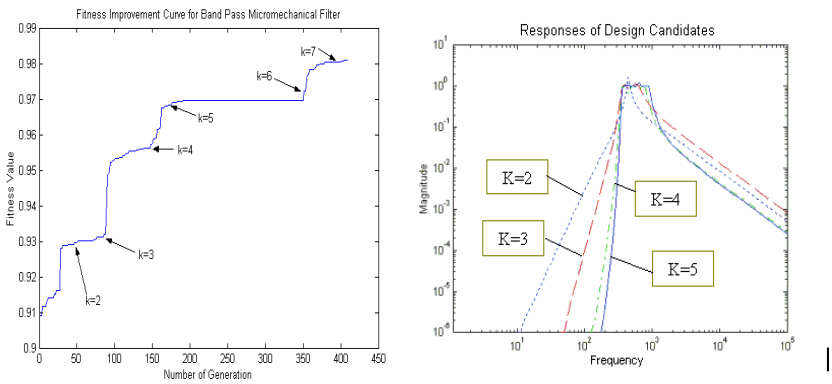

Fig. 14. (left) Fitness progress over generations (right) Frequency responses of design candidates at different generations.

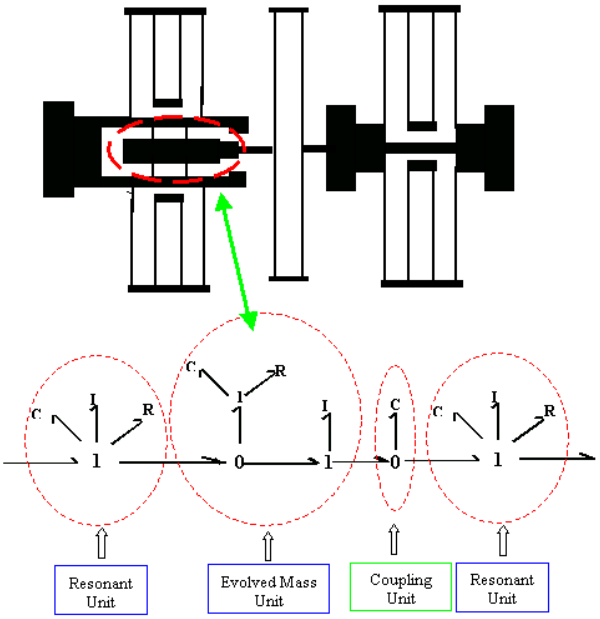

Fig. 15. A novel design topology using a semi-realizable function set.

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 set guarantees that the phenotypes generated can be built using existing or anticipated manufacturing technology. Large-scale component reuse and assembly

of MEMS is expected to show more applicability and promise of this method for MEMS design.

4.3 Synthesis of Suspension System Controllers

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. A good suspension design is concerned with disturbance rejection from these disturbances to the outputs (e.g. vertical position of vehicle mass), the basis for evaluating performance. In general, a suspension system needs to be "soft" to insulate against road disturbances and "hard" to insulate against load disturbances.

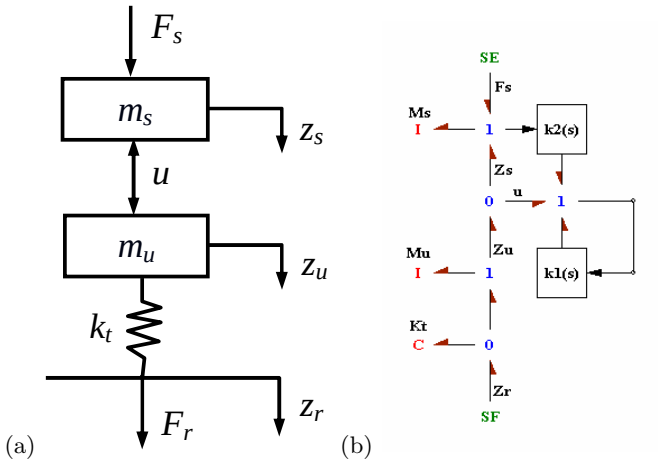

Fig. 16. (a) Quarter-car model in iconic diagram; (b)Quarter-car suspension control with both road and load disturbances.

A quarter-car iconic model is illustrated in Fig. 16(a). 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 multi-port 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:

$$\begin{bmatrix} F_r \\ \dot{z}_s \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} \dot{z}_r \\ F_s \end{bmatrix} \tag{3}$$

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.

$$m_s \ddot{z}_s = -u + F_s, \tag{4}$$

$$m_u \ddot{z}_u = u + k_t (z_r - z_u). \tag{5}$$

When load disturbance is also considered, the suspension system needs to be stiff to loads acting on the sprung mass. This requires in Eq. 3, $G_{11}(s)$ and $G_{21}(s)$ be set "soft" for road disturbance rejection while $G_{12}(s)$ and $G_{22}(s)$ be set "hard" for 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.16(b), with the control law taken to be: $u = \begin{bmatrix} k_1(s) & k_2(s) \end{bmatrix} \begin{bmatrix} z_s - z_u \\ s z_s \end{bmatrix}$, where $k_1(s)$ is collocated control, while $k_2(s)$ is non-collocated 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 specified in Eq.6. The desired load disturbance frequency response $H_2(s)$ is the frequency response specification for $G_{22}(s)$ in the immittance matrix in Eq.7. It is obtained by choosing certain suitable parameters in another double skyhook 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 $k_s = 150000N/m, c_1 = 12000Ns/m, c_2 = 6000Ns/m$. The desired $H_2(s)$ is calculated as:

$$H_1(s) = \frac{\dot{z}_s}{\dot{z}_r} = \frac{c_2 k_t s + k_s k_t}{m_s m_u s^4 + (c_1 m_u + c_2 m_s) s^3 + (k_s m_u + k_s m_s + k_t m_s) s^2 + c_1 k_t s + k_s k_t} \tag{6}$$

$$H_2(s) = \frac{\dot{z}_s}{F_s} = \frac{(m_u s^2 + c_2 s + k_t + k_s) s}{m_s m_u s^4 + (c_1 m_u + c_2 m_s) s^3 + (k_s m_u + k_s m_s + k_t m_s) s^2 + c_1 k_t s + k_s k_t} \tag{7}$$

GPBG Configuration

The design space of the controllers is bond graphs composed of C/I/R components and 1/0 junctions. We use the direct encoding formulation of the GPBG framework as specified in Section 3.3. There need be no embryo or modifiable site. The bond graphs are directly encoded by the GP trees. We use the following GP function set in Table 1. In this function set, the J0 and J1 functions have two inputs, meaning that in this encoding, the 1/0 junctions in the represented bond graphs can only have three ports: two input ports and one output port.

The fitness of a GP individual is evaluated by how accurately it approximates the desired frequency domain specification, to minimize

$\|dTF(j\omega) - tTF(j\omega)\|_2$,

where $dTF(j\omega)$ is the desired frequency response as specified by $H_1(s)$ and $H_2(s)$, and $tTF(j\omega)$is the theoretical frequency response of an evolved individual bond graph structure to be evaluated.

Design Experiments

Taking the desired road and load disturbance rejection responses $H_1(s)$and $H_2(s)$ as evaluation criteria, we used the following running setting for the experiments:

The best run of genetic programming using the basic function set in Table 1 produced the results shown in Fig.17 for $k_1(s)$, and Fig.18 for $k_2(s)$.

$$k_1(s) = \frac{2128s^3+46680s^2+1137000s+4792000}{s^2+16.08s+32.45} = \frac{2128(s+5.011)\left(s^2+16.93s+449.4\right)}{(s+2.366)(s+13.71)}$$

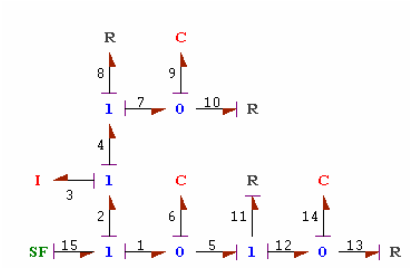

Fig. 17. Controller structure in bond graph form for $k_1(s)$

$$k_2(s) = \frac{10320s^3+453300s^2+40260000s+437000000}{s^3+172s^2+5799s+15890} = \frac{10320(s+12.04)\left(s^2+31.89s+3517\right)}{(s+3)(s+41.5)(s+127.5)}$$

Table 5. Experiment Settings

Objective:	Design a suspension system composed of two controllers.
Test fixture and embryo:	Two-input, two-output initial suspension system with a sprung mass, an unsprung mass, and a spring.
Program architecture:	Two result-producing GP species, k1 and k2, sharing the following attributes.
Function set:	For construction-continuing subtrees: $F_{ccs-rpb-initial}$ = {f0, f1, R, C, I}. For arithmetic-performing subtrees: F_{aps} = {ADD, SUB}.
Terminal set:	For arithmetic-performing subtrees: T_{aps} = {E}.
Fitness Cases:	41 frequency values in an interval of four decades of frequency values between 0.1Hz and 1,000Hz.
Raw Fitness:	Taking the desired road and load disturbance rejection responses as evaluation criteria, the raw fitness of a combined solution including individuals from both species is calculated as: $Fitness_{raw} = \sqrt{\frac{\sum_{i=1}^{n} (err_1+err_2)^2}{n}}$ n is the number of logarithmically sampled frequency points; err_1 and err_2 are the absolute differences of magnitude between the evolved and the desired road and load disturbance rejection frequency responses, respectively. $err_1 = \|G_{12}(j\omega) - G_{12}^{s}(j\omega)\|_2$; $err_2 = \left\|G_{11}(j\omega) - G_{11}^{h}(j\omega)\right\|_2$
Normalized Fitness:	$Fitness_{norm} = \frac{1.0}{Fitness_{raw}+1.0}$
Parameters:	Each species: 10 subpopulations of 100 individuals; Migration interval: 10 generations; Migration size: 2 individuals Crossover rate: 0.85; Mutation rate: 0.15; initializing tree depth: 2-4; maximum tree depth: 10-17
Result designation:	Best-so-far individual from max fitness species and matching individual from another species.
Termination:	When either species reaches max fitness value 0.99.

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, there is potential to discover good control strategies that may not be obtained through conventional methods.

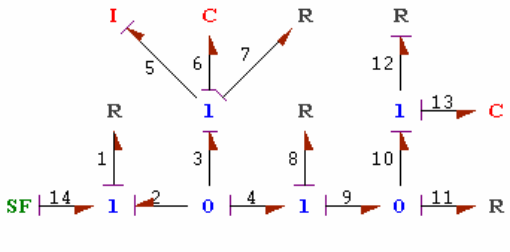

Fig. 18. Controller structure in bond graph form for $k_2(s)$

Fig.19 shows the simulation results as 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.

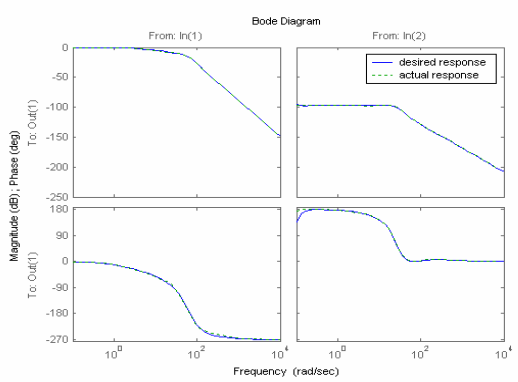

Fig. 19. Desired and actual responses of evolved suspension design Left (road disturbance response) Right (load disturbance response)

In summary, using the GPBG framework, we have evolved an active suspension system that 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.

4.4 Automatic Generation of Robust Designs

Although the topic of design for robustness cannot be addressed in detail in this chapter, application of GPBG for robust design has already been demonstrated. In [18], three strategies for using GPBG to synthesize robust passive analog filters were explored. The broadest conclusion was that filters of high robustness to variation in the values of their parameters could be evolved under GPBG, by introducing appropriate stochasticity during evolution of the topology of the filters. It did not require many more filter evaluations to evolve robust structures than to evolve those of similar nominal performance without stochasticity. It was also shown that robustness of designs with component values chosen from small, discrete sets could be improved by using only the "catalog" values during the evolutionary process, but adding stochastic variation about their nominal values.

5 Conclusions and Future Work

This chapter applies genetic programming and bond-graph system modeling – the GPBG approach – to topologically open-ended synthesis of engineering systems. Four real-world design problems have been examined, including mechanical vibration absorbers, MEMS filters, and suspension system controllers. Results illustrate that the GPBG framework is able to discover innovative designs that differ from those produced by human designers.

References

1. S. Ando and H. Iba. Linear genome methodology for analog circuit design. Technical report, Information and Communication Department, School of Engineering, University of Tokyo, 2000.
2. Z. Fan, J. Hu, K. Seo, E. D. Goodman, R. C. Rosenberg, and B. Zhang. Bond graph representation and GP for automated analog filter design. In E. D. Goodman, editor, *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 81–86, San Francisco, California, USA, 9-11 July 2001.
3. 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 E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 2058–2071, Chicago, 12-16 July 2003. Springer-Verlag.
4. C. Gagné and M. Parizeau. Open BEAGLE: A new versatile C++ framework for evolutionary computation. In E. Cantú-Paz, editor, *Late Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 161–168, New York, NY, July 2002. AAAI.

5. J. Hu and E. Goodman. Robust and efficient genetic algorithms with hierarchical niching and sustainable evolutionary computation model. In *Proc. 2004 Genetic and Evolutionary Computing Conference, Lecture Notes in Computer Science*, Chicago, 6 2004. Springer.
6. J. Hu, E. Goodman, and R. Rosenberg. Topological search in automated mechatronic system synthesis using bond graphs and genetic programming. In *Proc. of American Control Conference ACC 2004*, Boston, 7 2004.
7. N. Jalili. A comparative study and analysis of semi-active vibration-control systems. *Journal of Vibration and Acoustics*, 124:593, 10 2002.
8. D. Karnopp, D. L. Margolis, and R. C. Rosenberg. *System Dynamics: Modeling and Simulation of Mechatronic Systems.Third Edition.* John Wiley & Sons, Inc., New York, 2000.
9. J. R. Koza, D. Andre, F. H. Bennett III, and M. Keane. *Genetic Programming 3: Darwinian Invention and Problem Solving.* Morgan Kaufman, Apr. 1999.
10. J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane, and F. Dunlap. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 1(2):109–128, July 1997.
11. J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers, 2003.
12. J. R. Koza, M. A. Keane, J. Yu, F. H. Bennett III, and W. Mydlowec. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*, 1(1/2):121–164, Apr. 2000.
13. J. Lohn and S. Colombano. A circuit representation technique for automated circuit design. *IEEE Transactions on Evolutionary Computation*, 3(3):205–219, 1999.
14. H. M. Paynter. *An epistemic prehistory of bond graphs. In P. C. Breedveld and G. Dauphin-Tanguy (ed.), Bond Graphs for Engineers.* Elsevier Science Publishers, Amsterdam, 1991.
15. K. Seo, Z. Fan, J. Hu, E. D. Goodman, and R. C. Rosenberg. Dense and switched modular primitives for bond graph model design. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1764–1775, Chicago, 12-16 July 2003. Springer-Verlag.
16. K. Seo, Z. Fan, J. Hu, E. D. Goodman, and R. C. Rosenberg. Toward an automated design method for multi-domain dynamic systems using bond graph and genetic programming. *Mechatronics*, 13(8-9):851–885, 2003.
17. J. Wang and J. Terpenny. Integrated active and passive mechatronic system design using bond graphs and genetic programming. In B. Rylander, editor, *Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 322–329, Chicago, USA, 12–16 July 2003.
18. R. C. R. Xiangdong Peng, Erik D. Goodman. Comparison of robustness of three filter design strategies using genetic programming and bond graphs. In *Genetic Programming Theory and Practice (IV), Springer.Riolo, R., Soule, T. and Worzel, B. Eds.*, 2006.