PERGAMON

Mechatronics 13 (2003) 851–885

MECHATRONICS

Toward a unified and automated design methodology for multi-domain dynamic systems using bond graphs and genetic programming

Kisung Seo [a], Zhun Fan [a], Jianjun Hu [a], Erik D. Goodman [a,*], Ronald C. Rosenberg [b]

[a] *Genetic Algorithms Research and Applications Group (GARAGe), Michigan State University, East Lansing, MI 48824, USA*

[b] *Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824, USA*

Received 28 December 2001; accepted 18 July 2002

Abstract

This paper suggests a unified and automated design methodology for synthesizing designs for multi-domain systems, such as mechatronic systems. A multi-domain dynamic system includes a mixture of electrical, mechanical, hydraulic, pneumatic, and/or thermal components, making it difficult use a single design tool to design a system to meet specified performance goals. The multi-domain design approach is not only efficient for mixed-domain problems, but is also useful for addressing separate single-domain design problems with a single tool. Bond graphs (BGs) are domain independent, allow free composition, and are efficient for classification and analysis of models, allowing rapid determination of various types of acceptability or feasibility of candidate designs. This can sharply reduce the time needed for analysis of designs that are infeasible or otherwise unattractive. Genetic programming is well recognized as a powerful tool for open-ended search. The combination of these two powerful methods is therefore an appropriate target for a better system for synthesis of complex multi-domain systems. The approach described here will evolve new designs (represented as BGs) with ever-improving performance, in an iterative loop of synthesis, analysis, and feedback to the synthesis process. The suggested design methodology has been applied here to three design examples. The first is a domain-independent eigenvalue placement design problem that is tested for some sample target sets of eigenvalues. The second is in the electrical domain—design of analog filters to achieve specified performance over a given

* Corresponding author.
E-mail address: goodman@egr.msu.edu (E.D. Goodman).

0957-4158/03/$ - see front matter
doi:10.1016/S0957-4158(03)00006-0

frequency range. The third is in the electromechanical domain—redesign of a printer drive system to obtain desirable steady-state position of a rotational load.

Keywords: Automated design; Bond graph; Genetic programming; Multi-domain dynamic system

1. Introduction

Multi-domain dynamic system design differs from conventional design of electronic circuits, mechanical systems, and fluid power systems in part because of the need to integrate several types of energy behavior as part of the basic design. Multi-domain design is difficult because such systems tend to be complex and most current simulation tools operate over only a single domain. In order to automate design of multi-domain systems, such as mechatronic systems, a new approach is required [1]. The goal of the work reported in this paper is to develop a unified and automated procedure capable of designing mechatronic systems to meet given performance specifications, subject to various constraints. The most difficult aspect of the research is to develop a method that can explore the design space in a topologically open-ended manner, yet can find appropriate configurations efficiently enough to be useful and can be applied to multiple domains using a single tool. Our approach combines bond graphs (BGs) for representing the mechatronic system models with genetic programming (GP) as a means of exploring the design space.

BGs [2–6] allow us to capture the energy behavior underlying the physical aspects (as opposed to the information aspects) of mechatronic systems in a uniformly effective way across domains. They enable the analysis of multi-energy-domain systems with a unified inter-domain tool. Being topological structures, they are also ideal for representing a structured design space for open-ended generation and exploration. Finally, BGs allow efficient and rapid evaluation of individual designs, using a two-stage procedure—causal analysis of the graph followed, only if needed, by appropriate detailed calculation using a derived state model.

Sharpe and Bracewell [7] present the use of BG reasoning for the design of interdisciplinary schemes. They describe how conceptual scheme synthesis may be assisted and structured by the use of functions-mean trees developed by the application of BG-inspired rules. Youcef-Toumi [8] introduces an algorithm which identifies automatically the physical components and/or subsystems that are responsible for zero dynamics. Redfield [9] demonstrates the value of using BGs as a conceptual or configurational design tool for dynamic systems, using as an example a continuously variable transmission. Tay et al. [10] use a genetic algorithm to vary BG models. This approach adopts a variational design method, which means they make a complete BG model first, then change the BG topologically using a GA, yielding new design alternatives. Their goal is to provide a wider range of possible designs, and is closely related to that presented here, but within a topologically more limited search space.

GP is an effective way to generate design candidates in an open-ended, but statistically structured, manner. A critical aspect of the procedure is a fitness (or performance) measure, which must guide the evolution of candidate designs toward a suitable result in a reasonable time. There have been a number of research efforts aimed at exploring the combination of GP with physical modeling to find good engineering designs. Perhaps most notable is the work of Koza et al. [11–14]. He presents a single uniform approach using GP for the automatic synthesis of both the topology and sizing of a suite of various prototypical analog circuits, including low-pass filters, operational amplifiers and controllers. This system has already shown itself to be extremely promising, having produced a number of patentable designs for useful artifacts, and is the most closely related approach to that proposed here; however, it works in a single energy domain. That means his approach requires a different simulation code or tool for each application. Writing a simulation code for an application is a very time-consuming job. If the design applications or domains are different, one must write or link to a simulation code for each new application.

The approach described here includes the potential advantages of both BGs and GP, with a powerful synergistic effect for automated, multi-domain, and topologically open-ended design. We use a unified evaluation tool based on BGs, most of which can be used for every application, even if they are in different domains, with relatively minor supplemental codes to provide any additional functionality required.

In this paper, we have not attempted to duplicate the results of other researchers such as Koza et al. for a specific problem; rather, we have demonstrated the effectiveness of our design methodology for applications in each of several different domains. As our first class of design problems, we chose one in which the objective is to realize a design having a specified set of eigenvalues. Since the problem can be studied effectively using linear components with constant parameters, we only needed to introduce one-port (generalized) resistance, capacitance, and inductance elements in our designs. Section 2 discusses the inter-domain nature, efficient evaluation and graphical generation of BGs. Section 3 describes evolution of BGs by GP. Sections 4–6 presents some results for an eigenvalue design, electric filter design and printer drive redesign problem, and Section 7 concludes the paper.

2. A design methodology based on bond graph and genetic programming

2.1. Unified and automated methodology and multi-domain dynamic systems

Due to the complexity of the engineering design problem, the need for efficiency in the design methodology is greatly increased. The most critical issues are automation of the design process and use of a unified design tool. Most design tools or methodologies require user interaction, so users must make many decisions during the design process. This makes the design procedure more complex and often introduces the need for trial-and-error iterations. The other issue is the need for a unified design tool that can be applied to several domain areas—electrical, mechanical, hydraulic,

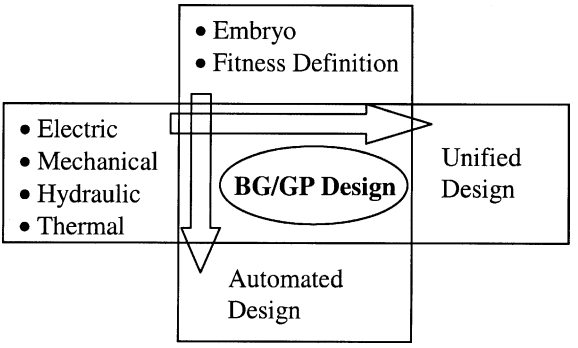

Fig. 1. Key features of the BG/GP design methodology.

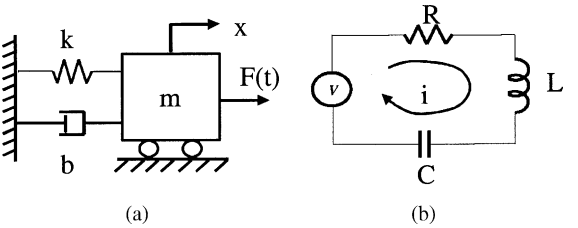

Fig. 2. Example single-domain systems: (a) mechanical, and (b) electrical.

etc. Designers sometimes have to consume large amounts of time to prepare new analysis tools or methods. A design methodology that combines BGs and GP can provide both capabilities—an automated and unified approach (Fig. 1). The proposed BG/GP (bond graph with genetic programming) design methodology requires only an embryo model and fitness (performance) definition in its initial stage; the remaining procedures are automatically executed by GP search.

Multi-domain system design differs from conventional design of electronic circuits, mechanical systems, and fluid power systems in part because of the need to integrate several types of energy behavior as part of the basic design. For example, in addition to appropriate "drivers" (sources), lumped-parameter dynamical mechanical systems models typically include at least masses, springs and dampers (Fig. 2(a)) while "RLC" electric circuits include resistors, inductors and capacitors (Fig. 2(b)). Fig. 3 shows a drive system for a printer that involves a drive shaft and a load, with important physical properties modeled. The input is the driving torque generated through the belt coupling back to the motor. Fig. 2 shows examples of single-domain systems, while Fig. 3 represents a mixed-domain system.

2.2. Bond graphs

The BG is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems. BG models can describe the dynamic behavior of

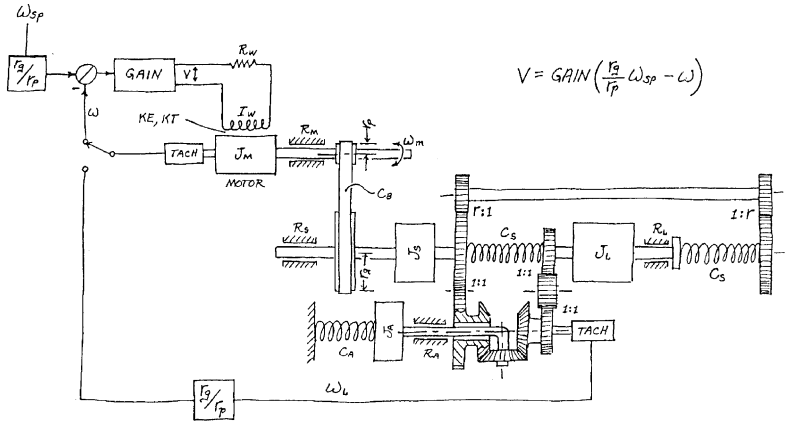

Fig. 3. Schematic diagram of an example mechatronic system—the printer drive.

physical systems by the connection of idealized lumped elements based on the principle of conservation of power.

BGs consist of elements and bonds. There are several types of elements, each of which performs analogous roles across energy domains. The first type—C, I, and R elements—are passive one-port elements that contain no sources of power, and represent capacitors, inductors, and resistors (in the electrical domain). The second type, S_e and S_f, are active one-port elements that are sources of power, and that represent effort sources and flow sources, respectively (for example, sources of voltage or current, respectively, in the electrical domain). The third type, TF and GY, are two-port elements, and represent transformers and gyrators, respectively. Power is conserved in these elements. A fourth type, denoted as 0 and 1 on BGs, represents junctions, which are three-port (or more) elements. They served to interconnect other elements into subsystems or system models. Some example BG models are shown below. Fig. 4 consists of S_e, 1-junction, C, I, and R elements, and that same BG represents either a mechanical mass, spring and damper system (Fig. 2(a)), or an RLC electric circuit (Fig. 2(b)). S_e corresponds with force in mechanical, voltage in electric. The 1-junction implies a common velocity for (1) the force source, (2) the end of the spring, (3) the end of the damper, and (4) the mass in the

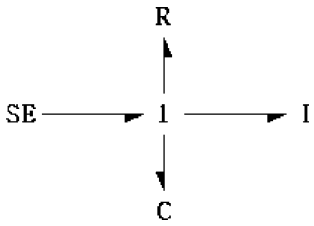

Fig. 4. BG model for Fig. 2(a) and (b).

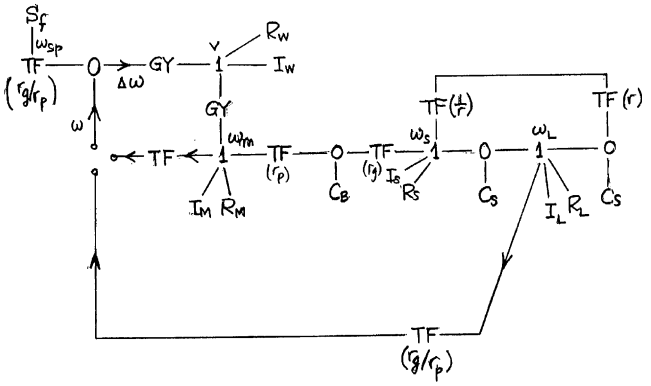

Fig. 5. BG model for printer drive of Fig. 3.

mechanical system, and implies that the current in the RLC loop is common. The R, I, and C represent the damper, inertia (of mass), and spring in the mechanical system, or the resistor, inductor, and capacitor in the electrical circuit.

Besides the basic R, I and C elements, two-port elements TF and GY are used in the printer drive system in Fig. 5. Transformers TF relate efforts to efforts and flows to flows, while gyrators GY relate the effort at one port to the flow at the other. In this model, TF corresponds to a gear ratio or signal ratio, while GY relates gain to voltage or current in a motor to mechanical rotation.

One of the important concepts in BG theory is *causality*. If two components are bonded together in a BG, we can think of one effort as causing one component to respond with a flow while the flow causes the first component to respond with an effort. Thus the cause–effect relations for efforts and flows are represented in opposite directions. A single mark on a bond, which is called the *causal stroke*, indicates how e and f simultaneously are determined causally on a bond (Fig. 6). This concept plays a great role in determining the feasibility of a design very simply at an early stage (see Section 3.3).

BGs have three embedded strengths for design applications—the wide scope of systems that can be created because of the multi- and inter-domain nature of BGs, the efficiency of evaluation of design alternatives, and the natural combinatorial features of bond and node components for generation of design alternatives. First, multi-domain systems (electrical, mechanical, hydraulic, pneumatic, thermal) can be modeled using a common notation, which is especially important for design of mechatronic systems. For example, the mechanical system and the electrical circuit

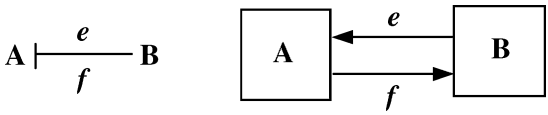

Fig. 6. The meaning of causal stroke.

in Fig. 2 have the *same* BG model (Fig. 4). Second, this representation of dynamic systems is also efficient for computational implementation. The evaluation stage is composed of two steps: (1) causality analysis, and, when merited, (2) dynamic simulation. In causality analysis, the causal relationships and power flow among elements and subsystems can reveal various system properties and inherent characteristics that can make the model unacceptable, and therefore make dynamic simulation unnecessary. While the strong typing used in the GP system (see below) will not allow the GP system to formulate "ill-formed" BGs, even "well-formed" BGs can have causal properties that make it undesirable or unnecessary to derive their state models or to simulate the dynamics of the systems they represent. Causality analysis is fast, and can rapidly eliminate further costs for many models that are generated by the GP system, by performing assignment of effort and flow variables and making checks for violations of the appropriate constraints. This simple filtering cuts the evaluation workload dramatically. For systems passing causal analysis, state equations are easily and systematically derived from BG models. Then various analyses (of eigenvalues, for example) or simulation, based on the state model, allow computation of the desired performance measures. Third, the graphical (topological) structure characteristic of BGs allows their generation by combination of bond and node components, rather than by specification of equations. This means that any system model can be generated by a combination of bond and node components, because of their free composition and unbounded growth capabilities. Therefore it is possible to span a large search space, refining simple designs discovered initially, by adding size and complexity as needed to meet complex requirements. The particular procedures used for synthesis of BG models are a developing and crucial part of this work, since they determine the search space within which design solutions will be contained.

2.3. Combining bond graphs with genetic programming

GP is an extension of the genetic algorithm, using evolution to optimize actual computer programs or algorithms to solve some task [15,16], typically involving a graph-type (or other variable-length) representation. The most common form of GP is due to John Koza [11–13], and uses trees to represent the entities to be evolved. GP can manipulate variable-sized strings and can be used to "grow" trees that specify increasingly complex BG models, as described below. If the scope and analysis efficiency of the BG model can be successfully integrated with the impressive search capability of GP when utilized to its full potential, an extremely capable automated synthesis procedure, without need for user intervention, should result.

As with any fairly general system for design automation, the user must, as part of the specification of the problem to be solved, indicate the target performance that is desired and how it is to be evaluated. That generally includes identifying some input variable(s) or driver(s) and some output(s) at which the desired behavior is to be observed, and the desired relationships among them. For a system to be represented as a BG, this amounts to specifying an "embryonic" physical model for the target system, which will remain *invariant* during the design process. That embryo should

include any exogenous inputs, usually specified as time-varying sources of effort or flow (e.g., voltages, currents, forces, velocities, pressures, etc.). It must include any outputs required to evaluate fitness (for example, voltages across a given load resistance, flow rates through pipes, etc.). That these components should NOT be allowed to be changed/eliminated during design evolution is obvious—the problem is not defined without their presence. When the user has formulated the problem (i.e., the external boundaries of the physical model with its environment and the performance measures to be used), the user must specify it as an embryonic BG model and a "fitness" function (objective function to be extremized). The user also specifies one or more "sites" in the embryo model where modifications/insertions are allowed. Then an initial population of trees is created at random, using that embryo as a common starting point. For each tree ("individual"), the BG analysis is performed. This analysis, including both causal analysis and (under certain conditions) state equation analysis, results in assignment of a fitness to the individual. Then genetic operations—selection, crossover and mutation—are performed on the evaluated population, generating new individuals (designs) to be evaluated. The loop, including BG analysis and GP operation, is iterated until the termination condition is satisfied. The result is one or more "best" BGs ready for physical realization (there is, of course, no basis for asserting the global optimality of any solution that arises—it is simply the best generated, and the procedure is considered successful if the quality of that design is adequate for the designer's purposes).

3. Evolutionary design with bond graphs

3.1. Bond graph construction

A typical GP system (like the one used here) evolves GP trees, rather than more general graphs. However, BGs can contain loops, so we do not represent the BGs directly as our GP "chromosomes". Instead, a GP tree specifies a *construction procedure* for a BG. BGs are "grown" by executing the sequence of GP functions specified by the tree, using the BG embryo as the starting point.

Initial studies were reported in Seo et al. [17] and Fan et al. [18]. The following set of BG elements: {C, I, R; 0, 1}, plus any sources in the embryo, are used in the studies reported here. This set is sufficient to allow us to achieve designs that have practical meaning in engineering terms, while still permitting other methods to be used for comparison, as an aid in assessment of our work.

We define the GP functions and terminals for BG construction as follows. There are four types of functions: first, *add* functions that can be applied only to a junction and which add a C, I, or R element; second, *insert* functions that can be applied to a bond and which insert a 0-junction or 1-junction into the bond; third, *replace* functions that can be applied to a node and which can change the type of element and corresponding parameter values for C, I, or R elements; and fourth, *arithmetic* functions that perform arithmetic operations and can be used to determine the numerical values associated with components (Table 1).

Table 1
GP terminals and functions

Name	#Args	Description
add_C	4	Add a C element to a junction
add_I	4	Add an I element to a junction
add_R	4	Add an R element to a junction
insert_J0	3	Insert a 0-junction in a bond
insert_J1	3	Insert a 1-junction in a bond
replace_C	2	Replace the current element with a C element
replace_I	2	Replace the current element with an I element
replace_R	2	Replace the current element with an R element
+	2	Add two ERCs
−	2	Subtract two ERCs
enda	0	End terminal for add element
endi	0	End terminal for insert junction
endr	0	End terminal for replace element
erc	0	ERC

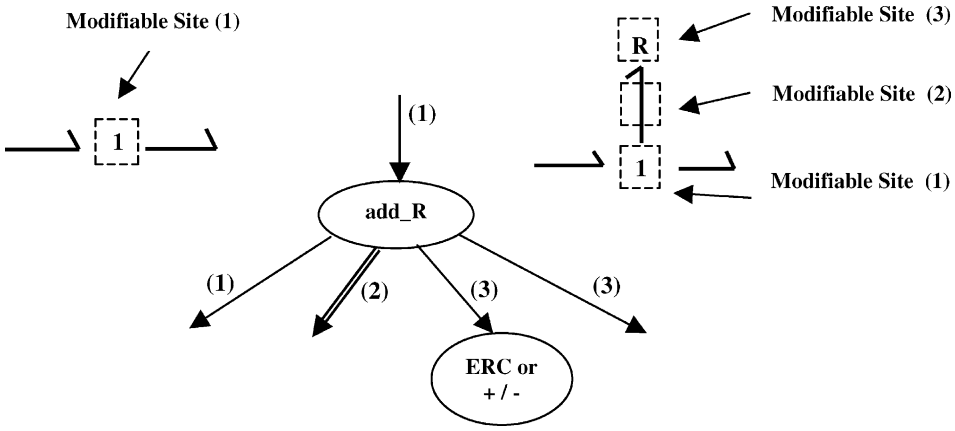

Fig. 7. The add_R function.

Some typical operations—add_R (a 1-port resistor) and insert_J0 (a 0-junction)—are explained in detail as follows. In Fig. 7, the R element is added to an existing junction by the add_R function. This function adds a node with a connecting bond. An R element also requires an additional parameter value (ERC—ephemeral random constant). The insert_J0 function can be applied only at a bond, and performs insertion of a 0-junction at the given modifiable site (Fig. 8). Inserting a 0-junction between node R and a 1-junction yields a new BG (the right side of Fig. 8). As a result, three new modifiable sites are created in the new BG. At each modifiable site,

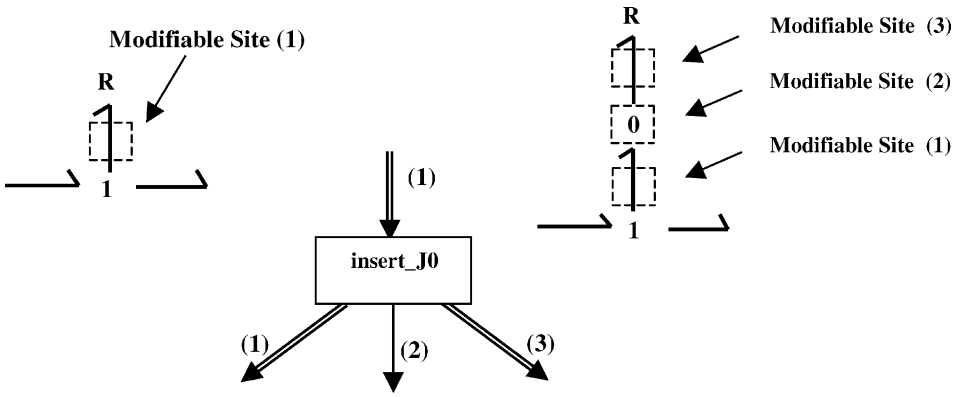

Fig. 8. The insert_J0 function.

various bond growth functions can be applied, in accordance with its type. In GP terminology, this is a *strongly typed* GP.

Fig. 9 shows an example of a GP tree, generated at random from the embryo root node. There are three modifiable embryo sites, denoted "1" (BG node), "a" (bond), and "2" (BG node). Each is denoted by an edge of the GP tree. If we follow edge 1

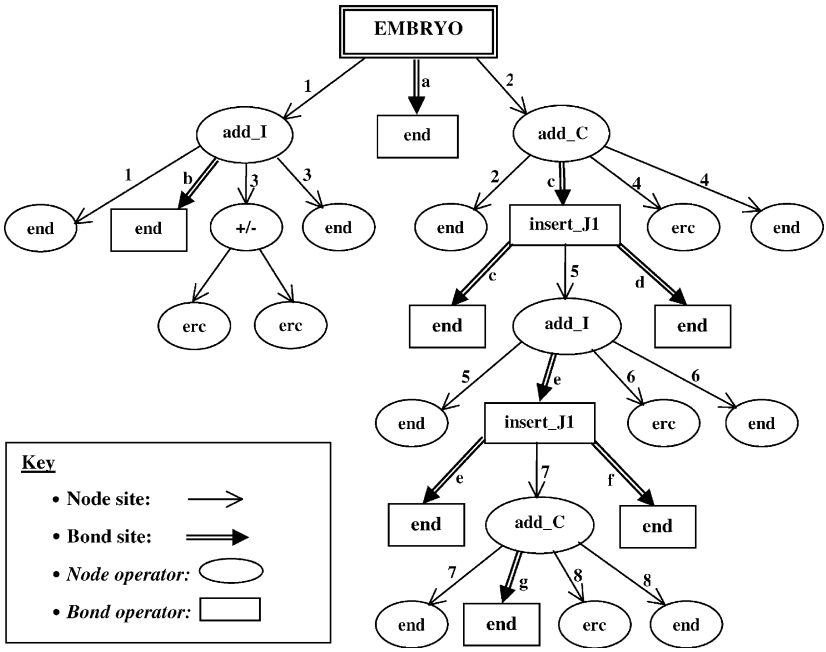

Fig. 9. Example of a GP tree.

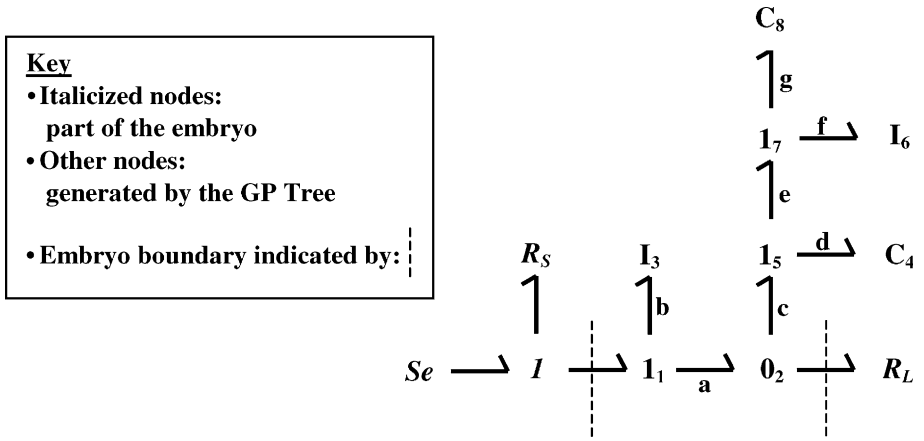

Fig. 10. BG generated by the example GP tree.

first, we see that an I element is added by add_I to the 1-junction (1_1) of the BG, I_3 in Fig. 10, together with its parameter value and a new bond. The result is to preserve modifiable site "1" and to add modifiable sites "b" and "3". The next set of operations under add_I in the GP tree show that all three sites happen to be made unmodifiable by appending of *end* functions.

Turning next to the edge labeled "a", we see that the first function applied to it is end. That bond site is thereby made unmodifiable. On the other hand, site "2" is the locus of additional BG growth. A C element is added by add_C to the 0-junction (0_2), C_4 in Fig. 10. For the next operation of insert_J1, a 1-junction (1_5) is inserted between the 0-junction (0_2) and C_4. After the remaining operations, the BG of Fig. 10 is generated from the GP tree of Fig. 9.

3.2. Overall design procedure

The flow of the entire algorithm is shown in Fig. 11. The user must specify the embryonic physical model for the target system (i.e., its interface to the external world, in terms of which the desired performance is specified). That determines an embryonic BG model and corresponding embryo (starting) element for a GP tree. From that, an initial population of GP trees is randomly generated. BG analysis is then performed on the BG specified by each tree. This analysis consists of two steps—causal analysis and (if justified) state equation analysis. Based on those two steps, the fitness function is evaluated. For each evaluated and sorted population, genetic operations—selection, crossover and mutation—are performed. This loop of BG analysis and GP operation is iterated until a termination condition is satisfied. The final step in instantiating a physical design would be to realize the highest-fitness BG in physical components, which is partially done (for electric filter) in the current work.

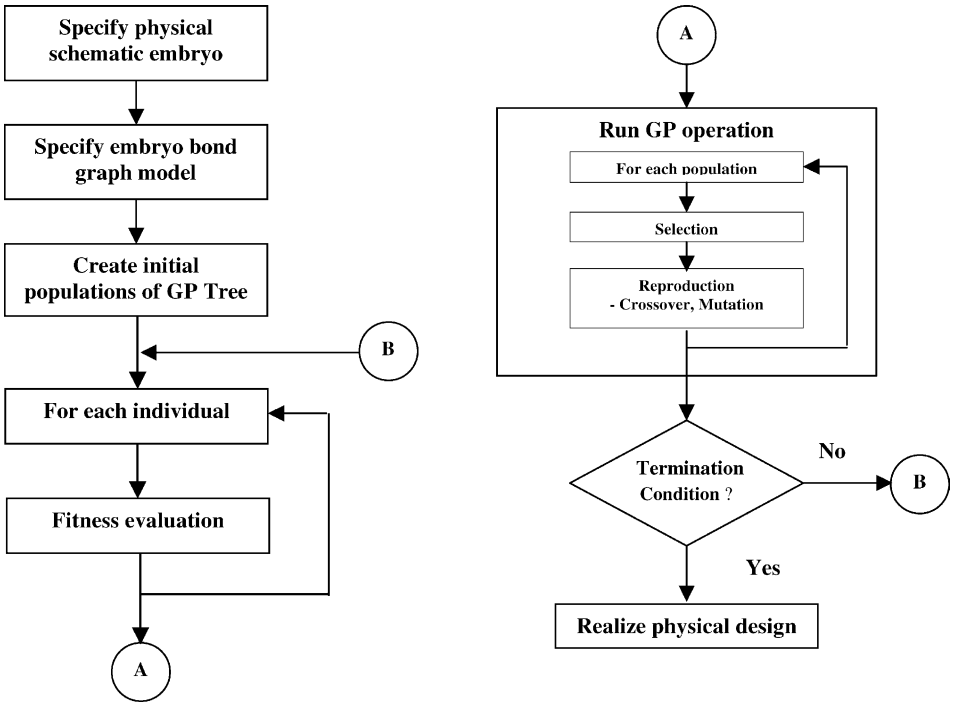

Fig. 11. The entire algorithm.

3.3. Bond graph evaluation

As mentioned earlier, a two-stage evaluation procedure is executed to evaluate BG models. The first, causal analysis [2], allows rapid determination of feasibility of candidate designs, thereby sharply reducing the time needed for analysis of designs that are infeasible. The causality assignment procedure is described as follows.

[Causality assignment procedure]

1. Choose any S_e or S_f, and assign its required causality. Immediately extend the causal implications, using all 0, 1, TF, and GY restrictions that apply.
2. Repeat step 1 until all sources have been causally assigned.
3. Choose any C or I and assign integral causality. Again extend the causal implications of this action, using all 0, 1, TF, and GY restrictions.
4. Repeat step 3 until all C and I elements have been causally assigned.
5. Choose any R that is unassigned and give it an arbitrary causality. Extend the causal implications of this action, using all 0, 1, TF, and GY.
6. Repeat step 5 until all R elements have been causally assigned.

Fig. 12 shows the example BG could have come from either an electric circuit or a mechanical schematic diagram. Starting from Fig. 12(a), step 1 is shown in Fig. 12(b), in which flow source (S_f) has been assigned its required causality. Fig. 12(c) shows the initial results of step 3. The element C2 now has its causal pattern, involving assignment of an effort to the 0-junction, so the causal mark on bond 3 must be as shown, according to permissible causal pattern of 0-junctions. By repeating step 3, I5 gains its causal pattern, leaving only one choice for bond 4 that gives the 1-junction a permissible causal pattern, as shown as Fig. 12(d).

For those designs "passing" the causal analysis, the state model is automatically formulated. The evaluation procedure is shown in Fig. 13.

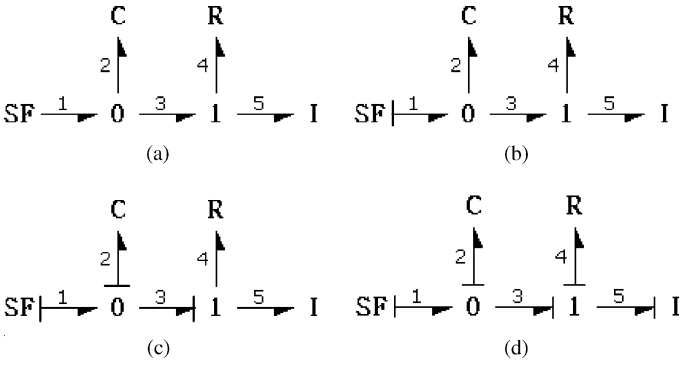

Fig. 12. Example of causality assignment.

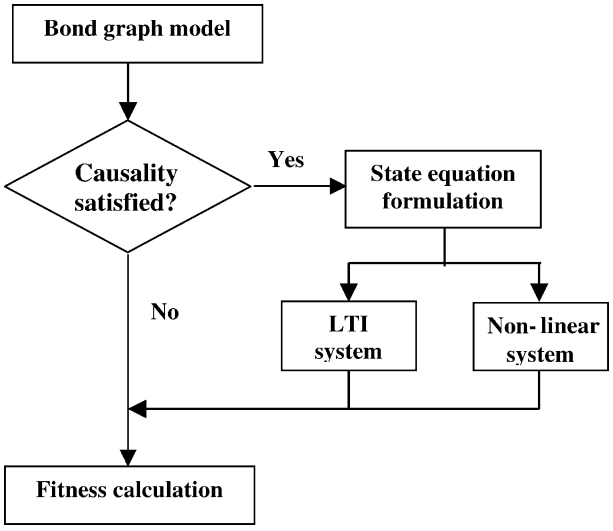

Fig. 13. Evaluation of BG models.

4. Case study 1—eigenvalue assignment

Although the final design of practical multi-domain systems still requires physical realization of the best generated BG model, it is sufficient to design a BG model with the desired performance in order to demonstrate the utility of our automated design methodology for multi-domain systems. In this work, the main design objective is to find BG models with minimal distance errors from the target sets of eigenvalues. The problem of eigenvalue assignment has received a great deal of attention in control system design. Design of systems to avoid instability and to provide specified response characteristics as determined by their eigenvalues is often an important and practical problem. The following experiments were done to illustrate the performance of GP on this problem and to explore the topological and parametric behaviors of the BG models evolved.

4.1. Problem definition

In the example that follows, a set of target eigenvalues is given and a BG model with those eigenvalues must be generated. An example of a set of target eigenvalues is shown in Fig. 14. The following three sets (consisting of two, four, and six target eigenvalues, respectively) were used as targets for example GP runs:

$$\{-1 \pm 2\mathrm{j}\}$$
$$\{-1 \pm 2\mathrm{j}, -2 \pm \mathrm{j}\}$$
$$\{-1 \pm 2\mathrm{j}, -2 \pm \mathrm{j}, -3 \pm 0.5\mathrm{j}\}$$

The fitness function is defined as follows: pair each target eigenvalue one:one with the closest one in the solution; calculate the sum of distance errors between each

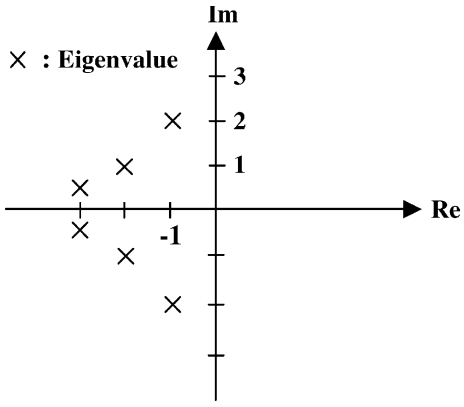

Fig. 14. An example of a target set of eigenvalues.

target eigenvalue and the solution's corresponding eigenvalue, divide by the order, and perform hyperbolic scaling as follows.

$$\text{Fitness (eigenvalue)} = 0.5 + 1\Big/\left(2 + \sum \text{error/order}\right)$$

The following sets of experiments (six total) were conducted, with each run repeated 10 times for each problem, all with different random seeds.

(1) Each of three target sets using an embryo with one modifiable site.
(2) Each of three target sets using an embryo with three modifiable sites.

The two types of embryo model used are shown in Fig. 15. Fig. 15(a) represents an embryo BG with one initial modifiable site, while the embryo BG in Fig. 15(b) has three initial modifiable sites. Each dotted box represents an initial modifiable site ("writehead" in GP parlance). In each case, the fixed components of the embryo are sufficient to allow definition of the system input and output, yielding a system for which the eigenvalues can be evaluated, including appropriate impedances. The construction steps specified in the GP tree are executed at that point. The numbers in parentheses represent the parameter values of the elements.

We used a strongly typed version Luke [19] of lil-gp [20] to generate BG models. These examples were run on a single Pentium III 1 GHz PC with 256 MB RAM. The GP parameters were as shown below.

Number of generations: 100 for two and four eigenvalues, 500 for six eigenvalues.
Population sizes: 100 in single population runs for two and four eigenvalues. 100 in each of 10 subpopulations for multiple population runs for six eigenvalues.
Initial population: half_and_half.
Initial depth: 2–6 for two and four eigenvalues, 3–6 for six eigenvalues.
Max depth: 17 for two and four eigenvalues, 12 for six eigenvalues (with 800 max_nodes).
Selection: tournament (size = 7).
Crossover: 0.9.
Mutation: 0.1.

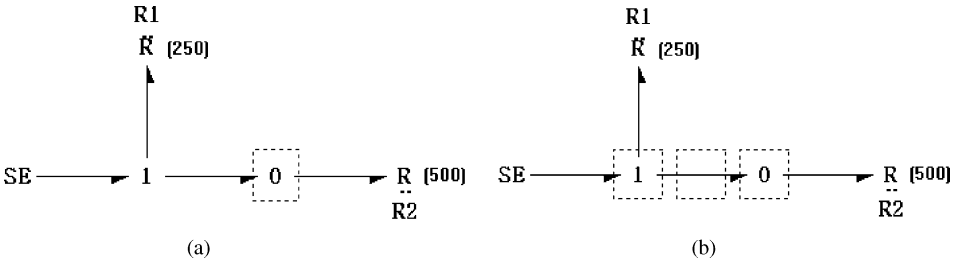

Fig. 15. Two types of embryo BG model with (a) one modifiable site, (b) three modifiable sites.

4.2. Results

Fig. 16 gives the solution eigenvalues obtained for a typical run with targets $-1 \pm 2j$, and summarizes the solutions, average distance errors from the targets, structure and parameter information. The corresponding BG model obtained is shown on the right side of Fig. 16. One 1-junction and three elements (one each of C, I, and R) were added to the embryo BG model of Fig. 15(a) in evolution of the solution. This final resulting BG is obtained after post-processing to remove unnecessary connections and reduce the non-state-variable R to its simplest equivalent form, using well-established rules. In the other runs, topologically similar structures, with C, I, and R elements and a 1-junction attached to the 0-junction, as in Fig. 16, dominated the results. The parameter value for R1 was 250 and R2 was 500, as shown in Fig. 15, for all experiments.

Fig. 17 illustrates solution eigenvalues and a corresponding BG model obtained for the same target set $-1 \pm 2j$, but starting from an embryo with three initial modifiable sites (Fig. 15(b)). Nonetheless, in this case, the C, I and R elements all evolved from the third modifiable site in the embryo, because only two state variables were needed. Nine of 10 runs had the same structure as Fig. 17. Only in one case were the C, I, and R attached to the 0-junction (first modifiable site).

Fig. 18 illustrates the solution eigenvalues obtained for the target set $-1 \pm 2j$, $-2 \pm j$, along with the corresponding BG model. One 1-junction and six elements (two each of C, I, and R) evolved from the embryo. Four state variables thus evolved—corresponding to each C or I element. In the four-eigenvalue problem, the topological search space is larger than in the two-eigenvalue problem, and a greater

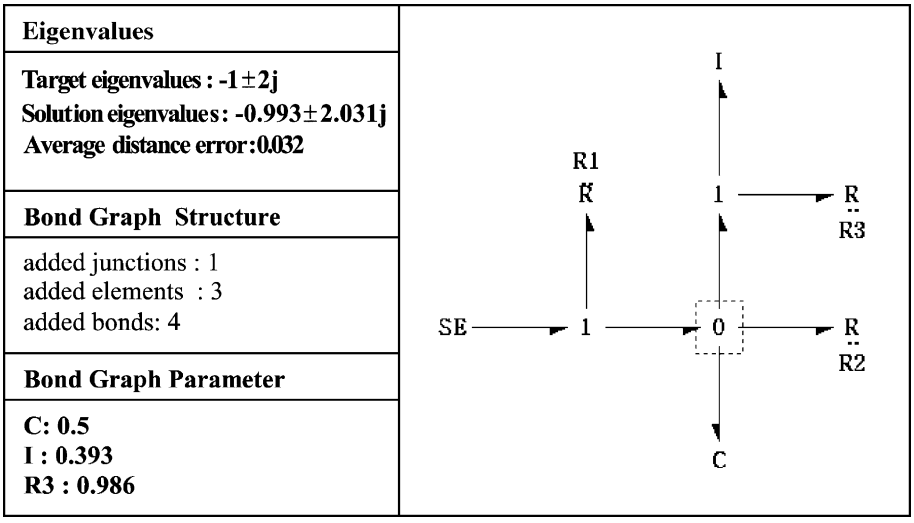

Fig. 16. Two-eigenvalues result with one modifiable site.

Eigenvalues
Target eigenvalues : -1±2j **Solution eigenvalues : -1.022±2.051j** **Average distance error : 0.056**
Bond Graph Structure
added junctions : 0 added elements : 3 added bonds: 3
Bond Graph Parameters
C: 0.215 **I : 0.885** **R3 : 24.191**

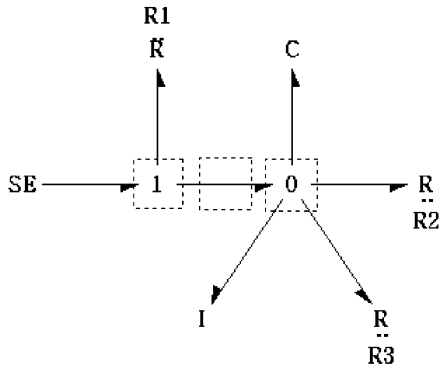

Fig. 17. Typical two-eigenvalue result from three modifiable sites.

Eigenvalues
Target eigenvalues : -1±2j, -2±j **Solution eigenvalues: -0.996±1.997j, -1.989±1.001j** **Average distance error: 0.008**
Bond Graph Structure
added junctions : 1 added elements : 6 added bonds: 6
Bond Graph Parameter
C1 : 0.029 **C2 : 1.189** **I 1 : 0.17** **I 2 : 6.822** **R3 : 13.179** **R4 : 0.209**

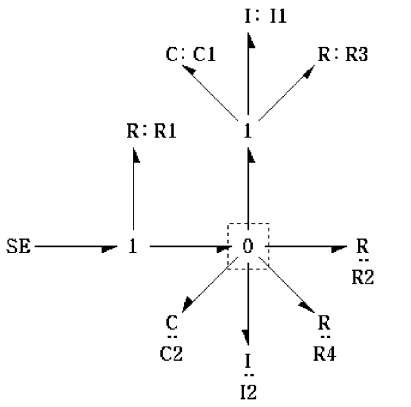

Fig. 18. Four-eigenvalue result from one initial modifiable site.

variety of structures is discovered. Half of them for the one modifiable site case have very similar structures to that of Fig. 18. It is interesting that in 3 of 10 cases, the numbers of C's and I's in the state vector are not the same (for example, some had one C and three I's). However, in most cases, especially with three modifiable sites, the C's and I's evolved in matched pairs.

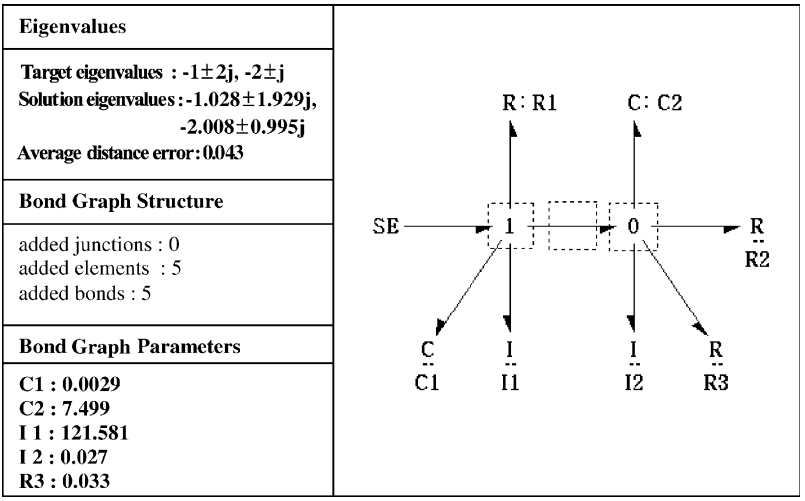

Fig. 19. Four-eigenvalue result from three initial modifiable sites.

Fig. 19 illustrates the result for the target set $-1 \pm 2j$, $-2 \pm j$ when started with three modifiable sites. Two C's, two I's and one R element evolved from the 0- and 1-junctions. Unlike the case of one modifiable site, components evolved at three different modifiable sites. In 8 cases of 10, the resulting BG appeared more balanced than when evolved from a single modifiable site.

The computation time for the six-eigenvalue problem was much larger than for the four-eigenvalue problem, It was more difficult to achieve acceptable error distances. It took 5–6 h on a Pentium III 1 GHz PC with 256 MB RAM for each solution. In order to get acceptable error distances, we used our hierarchical fair competition (HFC, Hu and Goodman [21]) method in the multi-population GP search. In order to reduce computation time, the GP parameters for max depth and max nodes were also reduced.

Figs. 20 and 21 illustrate two typical six-eigenvalue solutions for the target set $-1 \pm 2j$, $-2 \pm j$, $-3 \pm 0.5j$. Of course, both figures show that more junctions and elements were added, yielding a more complex structure than in the case of the four-eigenvalue problem. The solution starting from three modifiable sites has a more balanced structure, just as was found in the four-eigenvalue problem.

The tabular results of all runs are provided in Tables 2–4, including means, ranges, and standard deviations. Table 2 (two-eigenvalue problem) shows its relative ease of solution, with all runs producing quite accurate and similar results, from both one and three modifiable sites. Results in Table 3 (four-eigenvalue problem) show three runs from one modifiable site with relatively low fitnesses (under 0.95), while only one low-fitness run resulted from three modifiable sites. However, that difference is significant (t-test) only at the 0.2 level—this is still a relatively easy problem, and both embryos produced fairly good results.

Eigenvalues

Target eigenvalues: -1±2j, -2±j , -3±0.5j
Solution eigenvalues: -0.998±2.005j,
-1.983±0.961j
-2.970±0.492j
Average distance error: 0.02

Bond Graph Structure

added junctions : 4
added elements : 10
added bonds : 14

Bond Graph Parameter

C1 : 1.072, C2 : 2.702
C3 : 1.23
I 1 : 0.235, I2 : 0.982
I3 : 0.222
R3 : 0.289, R4 : 4.197
R5 : 0.106, R6 : 3.009

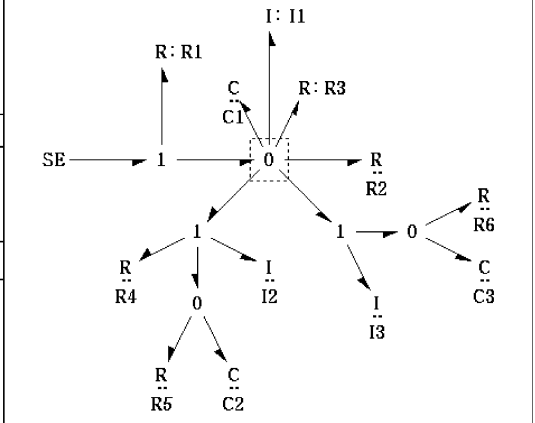

Fig. 20. Six-eigenvalue result with one initial modifiable site.

Eigenvalues

Target eigenvalues : -1±2j, -2±j , -3±0.5j
Solution eigenvalues: -1.014±2.012j,
-2.053±1.016j
-2.998±0.503j
Average distance error: 0.18

Bond Graph Structure

added junctions : 3
added elements : 10
added bonds : 13

Bond Graph Parameter

C1 : 0.106, C2 : 0.309
C3 : 0.000
I 1 : 100.574, I2 : 10.0
I3 : 570.908
R3 : 21.39, R4 : 40.393
R5 : 1.116, R6 : 0.002

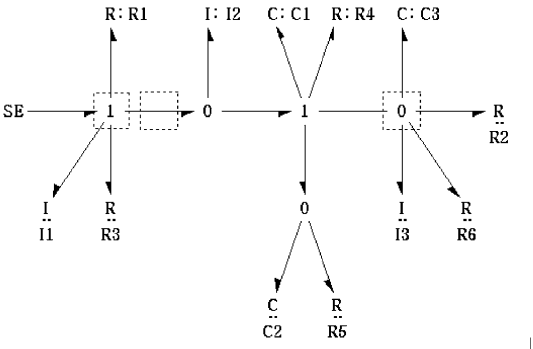

Fig. 21. Six-eigenvalue result with three initial modifiable sites.

Table 4 (six-eigenvalue problem) shows average distance errors that are much larger—this is a more difficult problem. The difference between embryos (number of modifiable sites) is clearer. Six of the runs from one modifiable site yielded fitness values under 0.95, but only one run from three modifiable sites did. The difference in the mean values between one-modifiable-site and three-modifiable-site runs was statistically significant at the 0.01 level (*t*-test). It appears that the added structural

Table 2
Summary of statistic result for two eigenvalues

	Fitness of two eigenvalues			
	One modifiable site		Three modifiable sites	
	Fitness	Average distance error	Fitness	Average distance error
1	0.999586	0.002	0.995738	0.017
2	0.999497	0.002	0.999198	0.003
3	0.999324	0.003	0.997157	0.011
4	0.994872	0.021	0.999634	0.001
5	0.999027	0.004	0.991560	0.034
6	0.979689	0.085	0.997047	0.012
7	0.999491	0.002	0.988743	0.046
8	0.996527	0.014	0.972420	0.117
9	0.995441	0.018	0.994286	0.023
10	0.999815	0.001	0.998803	0.005
Best	0.999815	0.001	0.999634	0.001
Worst	0.979689	0.085	0.972420	0.117
Average	0.996327	0.015	0.993459	0.027
SD	0.006136	0.026	0.008169	0.035

Table 3
Summary of statistic results of four eigenvalues

	Fitness of four eigenvalues			
	One modifiable site		Three modifiable sites	
	Fitness	Average distance error	Fitness	Average distance error
1	0.855715	0.811	0.905652	0.465
2	0.996989	0.012	0.998761	0.005
3	0.990091	0.040	0.992377	0.031
4	0.997378	0.011	0.994307	0.024
5	0.856205	0.807	0.993230	0.027
6	0.997716	0.009	0.993234	0.027
7	0.897575	0.515	0.992516	0.042
8	0.990289	0.040	0.993327	0.027
9	0.992603	0.030	0.996578	0.016
10	0.997442	0.010	0.995161	0.020
Best	0.997716	0.009	0.998761	0.005
Worst	0.855715	0.811	0.905652	0.465
Average	0.957200	0.229	0.985514	0.068
SD	0.061412	0.343	0.028132	0.140

flexibility of three modifiable sites makes it easier to reach a given target specification. It becomes more reasonable that the proper selection of modifiable sites in the embryo BG can affect the performance of the design evolved. Fig. 22 shows the fitness history of a typical six-eigenvalue run.

These results illustrate two things: (1) many topological forms of BG are capable of satisfying the specified design objectives, and (2) the form of embryo and GP

Table 4
Summary of statistic results for six eigenvalues

	Fitness of six eigenvalues			
	One modifiable site		Three modifiable sites	
	Fitness	Average distance error	Fitness	Average distance error
1	0.900513	0.497	0.987847	0.050
2	0.910099	0.438	0.967237	0.140
3	0.894659	0.534	0.967537	0.139
4	0.969219	0.131	0.969881	0.128
5	0.872007	0.688	0.973010	0.114
6	0.962446	0.162	0.912933	0.422
7	0.958229	0.182	0.965516	0.148
8	0.967667	0.138	0.963455	0.158
9	0.901091	0.493	0.965012	0.150
10	0.913675	0.417	0.952665	0.209
Best	0.969219	0.131	0.987847	0.050
Worst	0.872007	0.688	0.912933	0.422
Average	0.924961	0.368*	0.962509	0.166*
SD	0.035797	0.335	0.019508	0.098

* Indicates difference significant at 0.01 level (*t*-test).

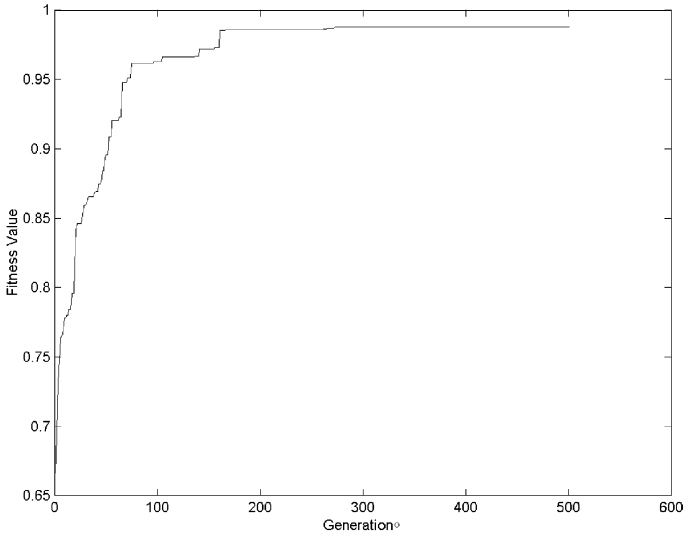

Fig. 22. Fitness history of a typical six-eigenvalue run.

operations used can strongly influence the form of the design evolved. Therefore, care and understanding of the evolutionary process are important in generating

designs that maintain desirable topological properties while satisfying the specified design objectives.

Although the experiments run to date are not sufficient to allow making strong statistical assertions, it appears that the search capability of GP is good enough to make feasible the automated design methodology proposed here for multi-domain systems.

5. Case study 2—analog filter design

5.1. Problem definition

A filter design problem was used as a test of our approach for evolving electrical circuits with BGs, as first reported in Fan et al. [18]. Three kinds of filters were chosen to verify our approach—high-pass, low-pass, and band-pass filters. The embryo electric circuit and corresponding embryo BG model used in our filter design are shown in Fig. 23. We used converted Matlab routines to evaluate frequency response of the filters created. As Matlab provides many powerful toolboxes for engineering computation and simulation, it facilitates development of source codes for our GP evaluation dramatically. In addition, as all individual circuits passed to Matlab code for evaluation are causally valid, the occurrence of singularities is excluded, which enables the program to run continuously without interruption. The fitness function is defined as follows: within the frequency range of interest, uniformly sample 100 points; compare the magnitudes of the frequency response at the sample points with

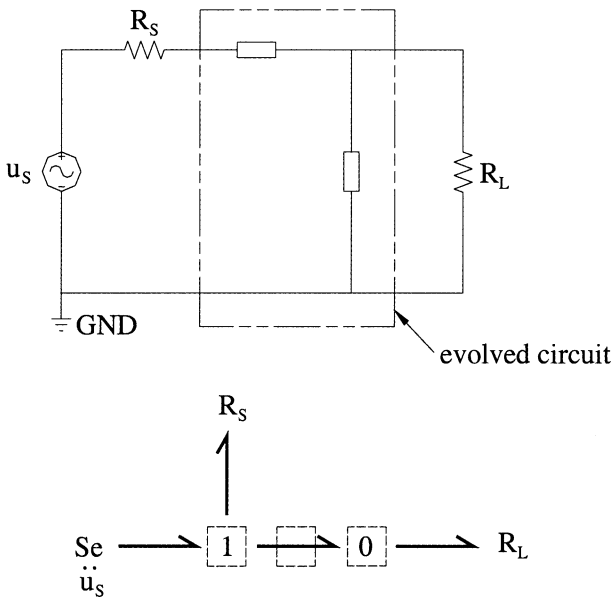

Fig. 23. Embryo circuit and its BG representation.

target magnitudes; compute their differences and obtain the squared sum of differences as raw fitness. Then normalized fitness is calculated according to:

$$\text{Fitness (filter)} = 100 \Big/ \left(100 + \sum \text{error}\right)$$

The GP parameters used for eigenvalue design were as follows:

Number of generations: 100.
Population size: 300 in each of 13 subpopulations and 2500 in each of 2 subpopulations for HFC.
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.

5.2. Results

To illustrate an intermediate step in the evolution of a high-pass filter with a target cutoff frequency of 1000 Hz, the performance of the best design evolved at generation 10 is shown in Fig. 24. It is clear that this design is far inferior to that evolved by the end of the run (fewer than 100 generations), as shown in Fig. 25.

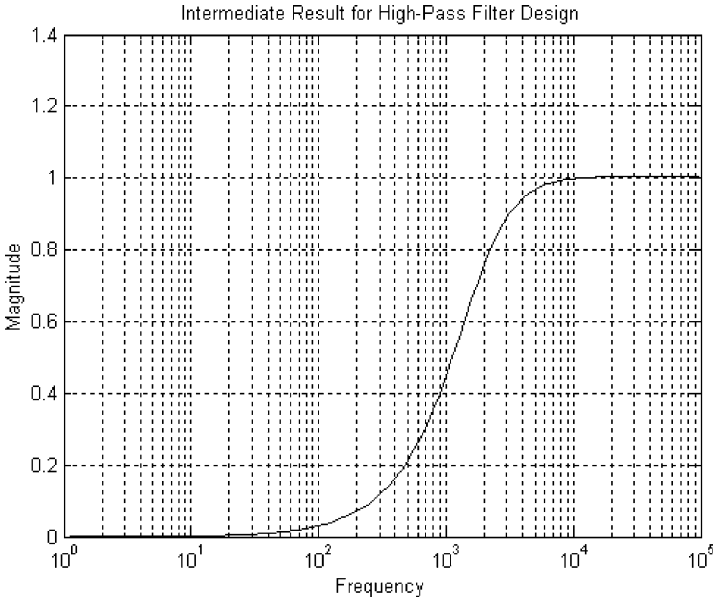

Fig. 24. Frequency response of intermediate high-pass filter.

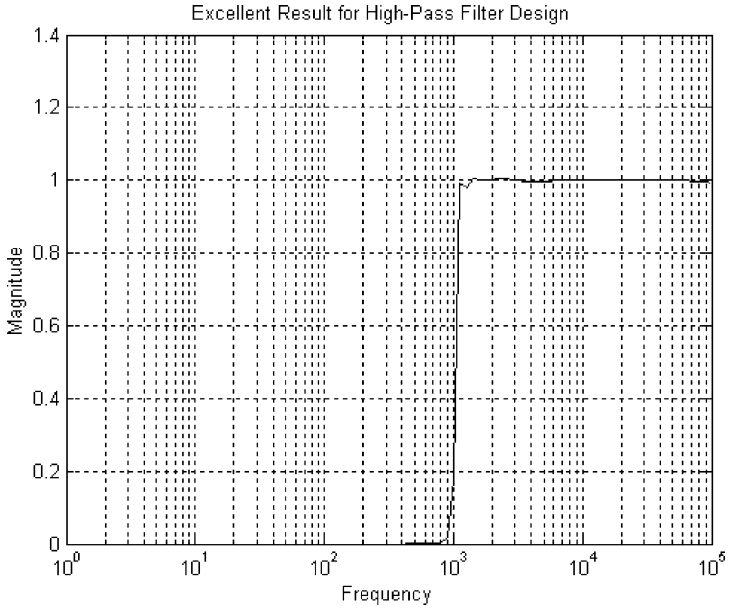

Fig. 25. Frequency response of evolved high-pass filter.

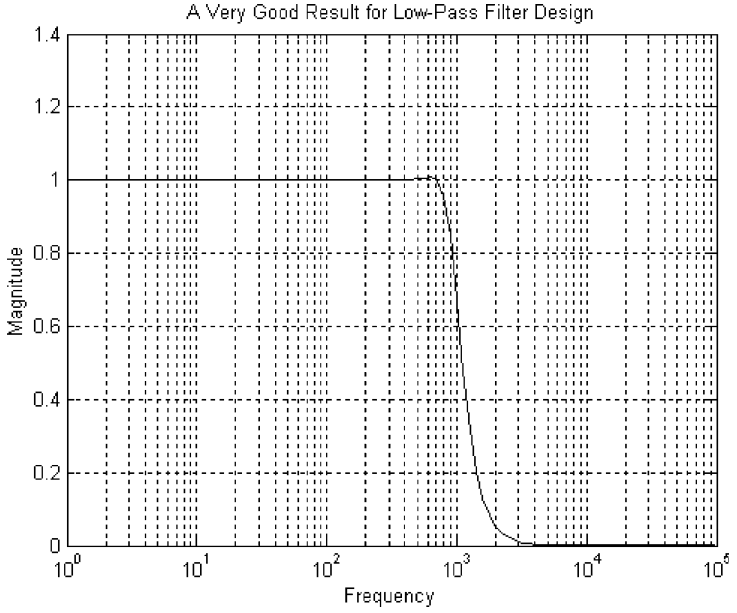

Fig. 26. Frequency response of evolved low-pass filter.

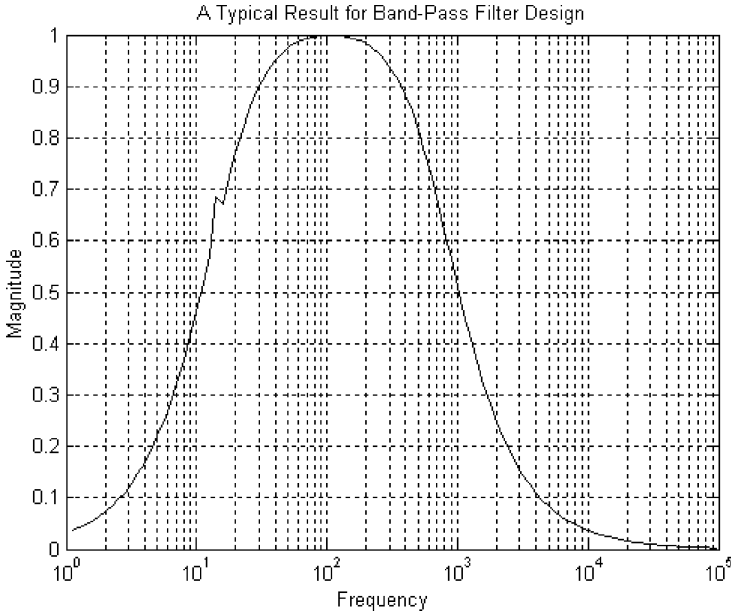

Fig. 27. Frequency response of evolved band-pass filter.

Fig. 26 gives the frequency response of an evolved low-pass filter with the same cutoff frequency. It shows that this result is also quite satisfactory. Fig. 27 gives the frequency response of an evolved band-pass filter with cutoff frequencies at 10 and 1000 Hz. Obviously, it is the most difficult of the three filter design problems. The evolved high-pass filter circuit and BG are shown in Figs. 28 and 29.

The statistical results of 10 runs each for high-, low- and band-pass filters are shown in Table 5. The distance errors between ideal frequency output and the output obtained, together with fitness values, are summarized. With the exception of some of the band-pass results, most were quite acceptable. Fig. 30 shows the fitness history of a typical high-pass filter run.

6. Case study 3—printer drive redesign

6.1. Problem definition

This example involves a drive system for a printer. The original problem was presented to one of the investigators by Denny and Oates of IBM, Lexington, KY, in 1972. Fig. 3 (in Section 2) shows a closed-loop control system to position a rotational load (inertia) denoted as J_L, and Fig. 31 shows the subsystem initially designed (manually). The detailed specification involved reducing the vibration of the load to an acceptable level, given certain command conditions for input position.

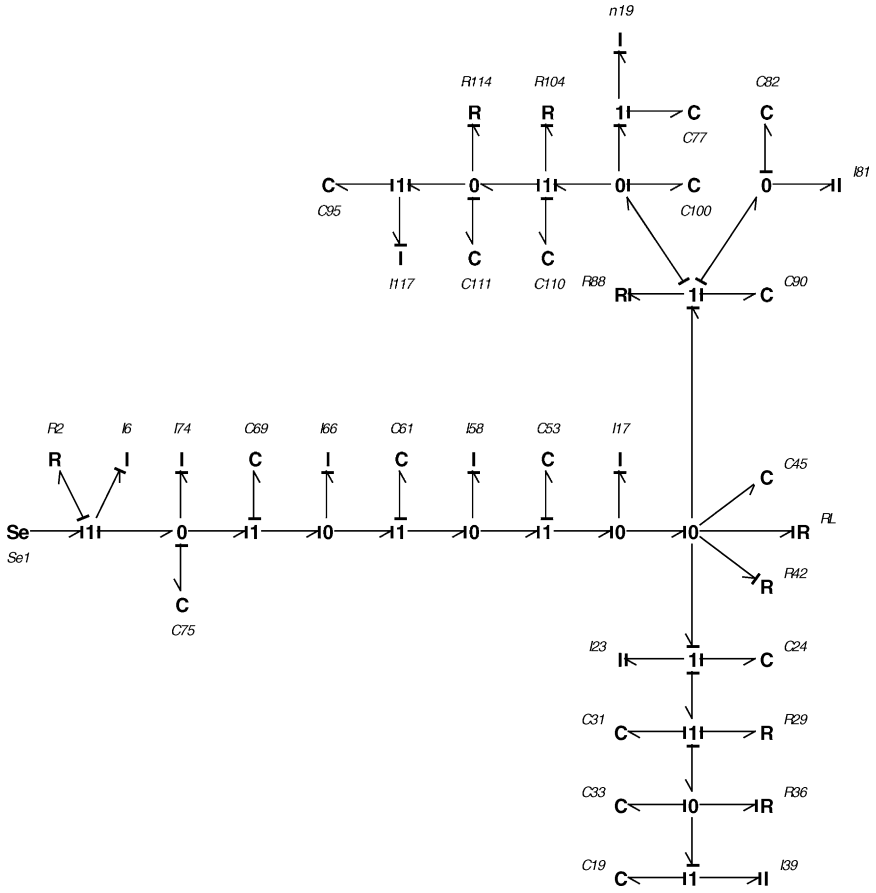

Fig. 28. BG representation for evolved high-pass filter.

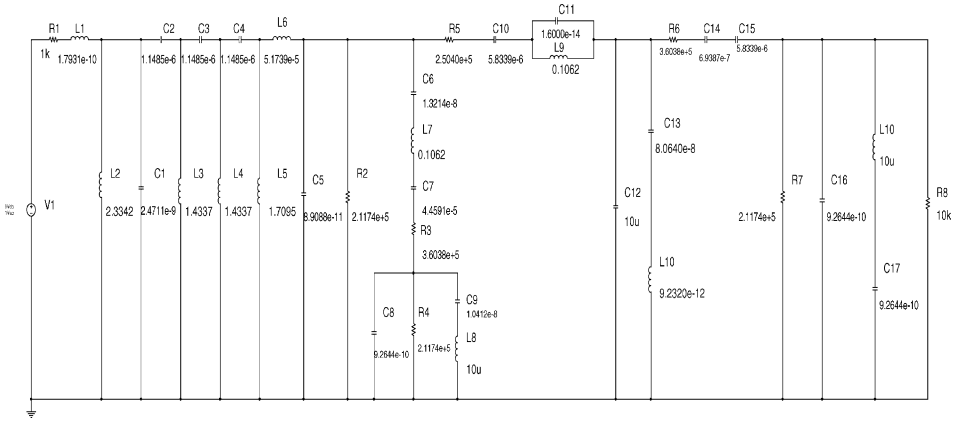

Fig. 29. Electric circuit for evolved high-pass filter.

Table 5
Summary results (errors, fitnesses) for filter designs

	Low-pass		High-pass		Band-pass	
	Error	Fitness	Error	Fitness	Error	Fitness
1	2.334	0.977188	3.349	0.967597	9.067	0.916868
2	3.428	0.966854	2.031	0.980089	12.861	0.886049
3	2.202	0.978455	1.159	0.988547	12.698	0.887325
4	3.032	0.970569	2.337	0.977163	12.672	0.887533
5	2.162	0.978838	0.828	0.991784	8.662	0.920282
6	3.427	0.966869	2.860	0.972199	12.864	0.886020
7	3.026	0.970633	3.287	0.968177	13.100	0.884177
8	2.951	0.971338	0.725	0.992797	13.090	0.884253
9	2.154	0.978914	1.141	0.988723	6.003	0.943373
10	1.988	0.980507	1.917	0.981192	13.049	0.884573
Best	1.988	0.980507	0.725	0.992797	6.003	0.943373
Worst	3.427	0.966869	3.349	0.967597	13.100	0.884177
Average	2.670	0.974017	1.963	0.980827	11.407	0.898045
SD	0.530	0.00502	0.936	0.008994	2.541	0.021033

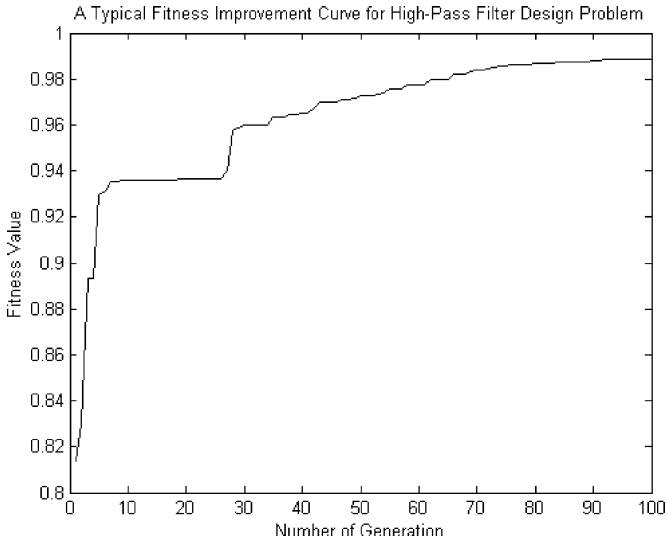

Fig. 30. Fitness history for a typical high-pass filter run.

The fitness function is defined as follows: within the time range of interest, uniformly sample 1000 points; compare the magnitudes of the step response at the sample points with target magnitudes; compute their differences and obtain the squared sum of differences as raw fitness. Then normalized fitness is calculated according to:

$$\text{Fitness (printer)} = 1000 \Big/ \left(1000 + \sum \text{error}\right)$$

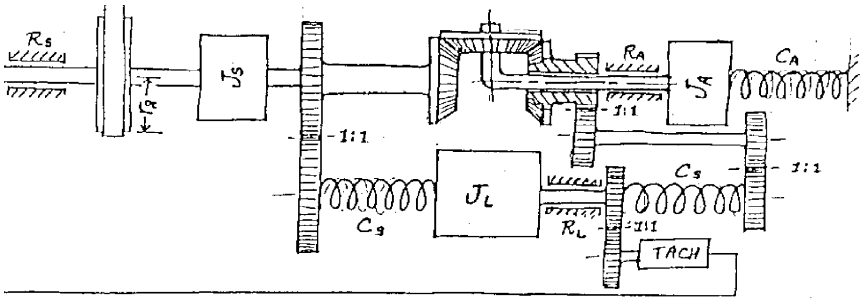

Fig. 31. The initial printer drive subsystem.

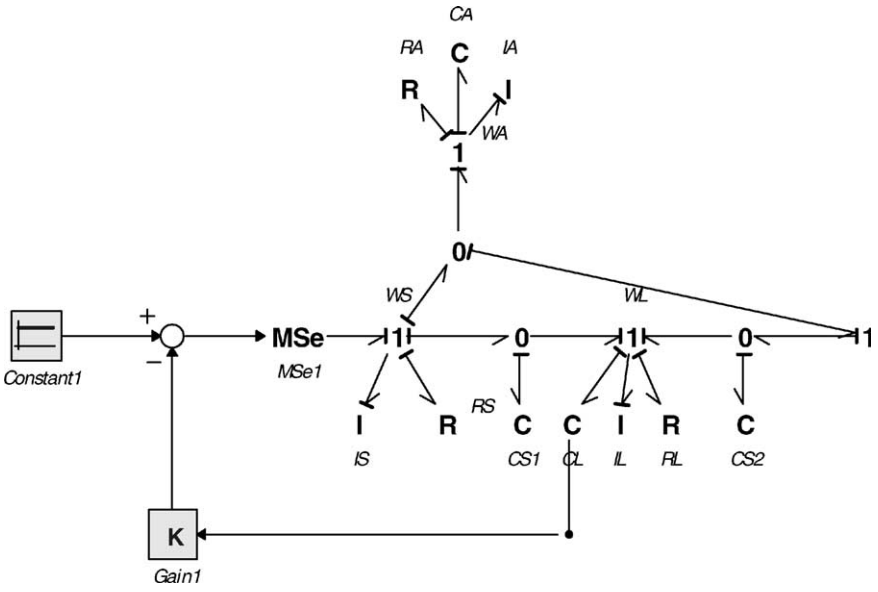

Fig. 32. The BG model for initial printer drive subsystem.

For comparison purposes, the performance simulation was first executed for the initial printer drive system. The corresponding BG model is shown in Fig. 32. The simulation result using the 20SIM software is as shown in Fig. 33. The input is a step function and feedback gain is one ($K = 1$). The output represents the position of the rotational load (J_L). This design is found to be unsatisfactory because of unacceptable vibration of the load for 1700–1800 ms.

To search for a new design using the BG/GP design tool, an embryo model is required. Fig. 34 shows an embryo subsystem that involves the drive shaft and the load, with important physical properties modeled. The input is the driving torque, T_d, generated through the belt coupling back to the motor (not shown). The subsystem is open-loop, with the feedback path disconnected. Since the vibration

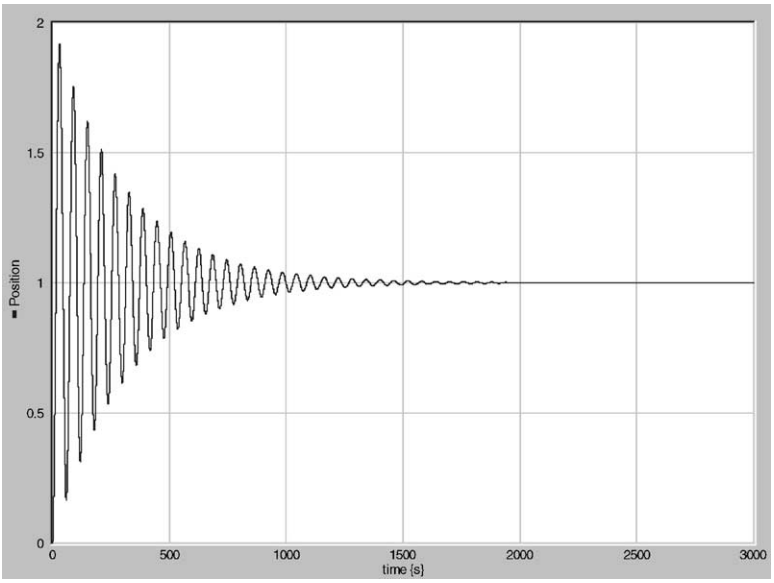

Fig. 33. Simulation result of initial printer drive subsystem.

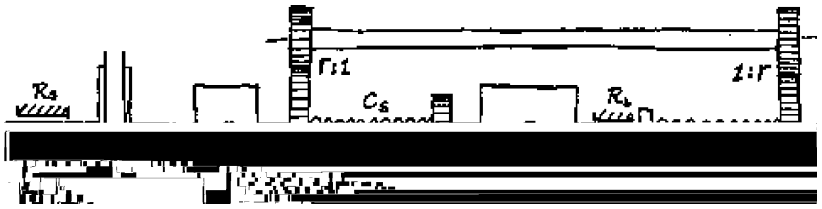

Fig. 34. The initial printer drive subsystem.

problem seemed to be local, this subsystem was deemed a logical place to begin the design problem.

The embryo model essentially constitutes a boundary condition for the design to be developed. The corresponding embryo BG model of the drive subsystem is given in Fig. 35. The two 1-junctions denote the angular velocities of the shaft inertia (w_S) and the load inertia (w_L), respectively. Also critical to the search procedure is the identification of sites in the model where modifications are permitted. We allowed three such sites, denoted by 1, 2, and 3 in circles. Sites 1 and 3 permit addition to the model; site 1 is essentially the rigid drive shaft section (w_S), and site 3 is at the right end of the drive shaft spring. Site 2 is an insertion location, where the connecting shaft can be "broken" and other subsystem effects inserted.

The parameters for the embryo model are:

I_s: 6.7×10^{-6} kg m^2,
R_s: 0.013×10^{-3} N m s/rad,

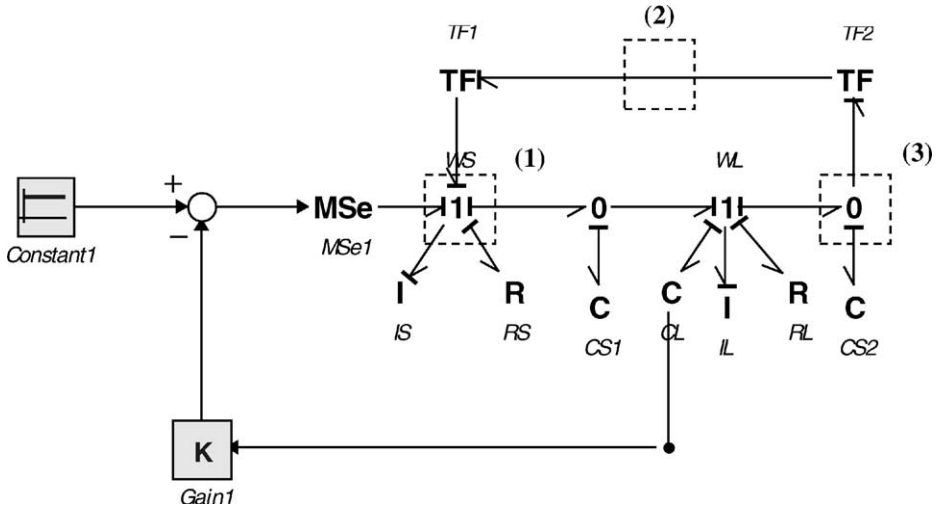

Fig. 35. The embryonic BG model with feedback loop.

C_{s1}: 0.208 N m/rad,
C_{s2}: 0.208 N m/rad,
R_L: 0.58×10^{-3} N m s/rad,
I_L: 84.3×10^{-6} kg m^2.

The following cases were run on a single Pentium III 1 GHz PC with 256 MB RAM. The GP parameters were as shown below.

Number of generations: 100.
Population sizes: 200 in each of 15 subpopulations for multiple population runs.
Initial population: half_and_half.
Initial depth: 3–6.
Max depth: 17.
Selection: tournament (size = 7).
Crossover: 0.9.
Mutation: 0.1.

6.2. Results

Ten runs of this problem have been done and most of the runs produced very good solutions. Two competing design candidates with different topologies, as well as their performances, are provided in Figs. 36–40 (evolved components are circled). We can see from the output rotational position responses that they all satisfy the design specification of settling time less than 70 ms. Note that the time scale of the plots is 100 ms.

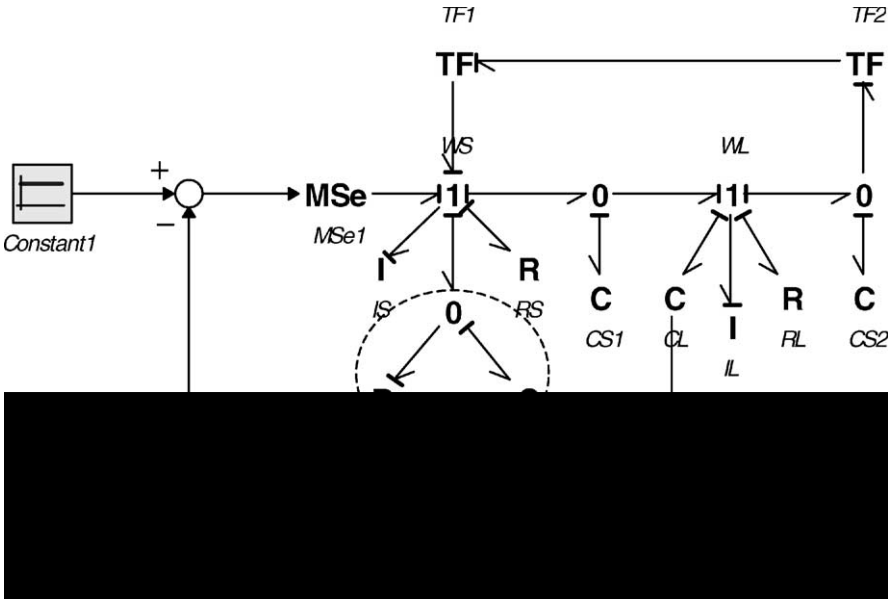

Fig. 36. The evolved BG model 1.

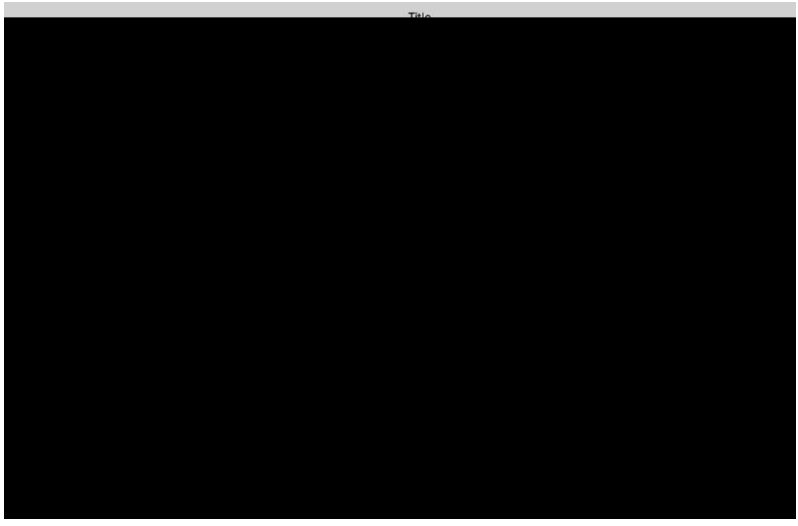

Fig. 37. Simulation result of evolved BG model 1.

One of the designs is shown in Fig. 36. It is generated in only 20 generations with 200 designs in each of 15 subpopulations, and has a very simple structure. Three elements, one each of 0-junction, C, and R, are added to modifiable site 1 of the

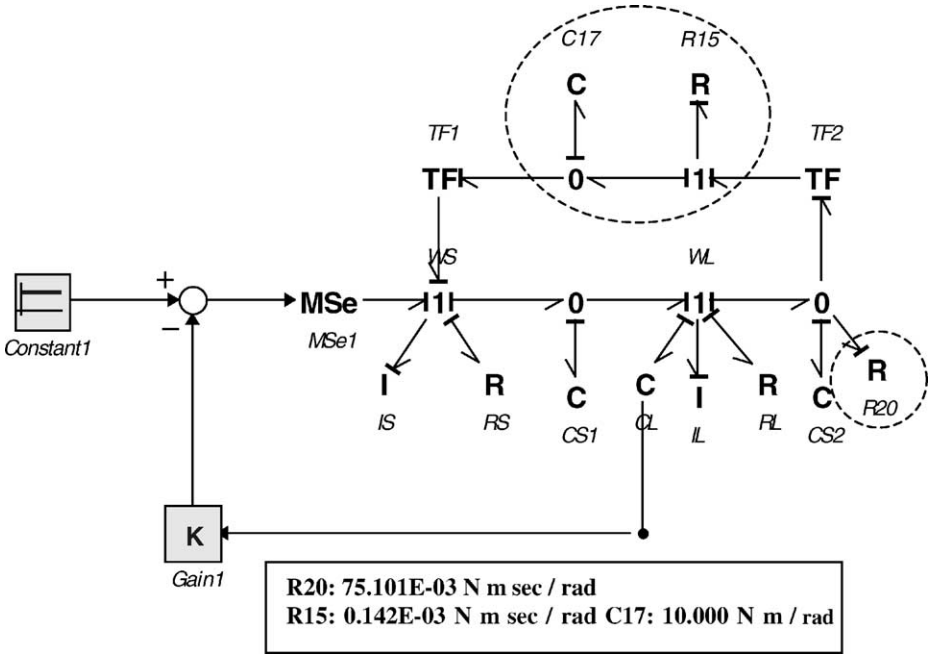

Fig. 38. The evolved BG model 2.

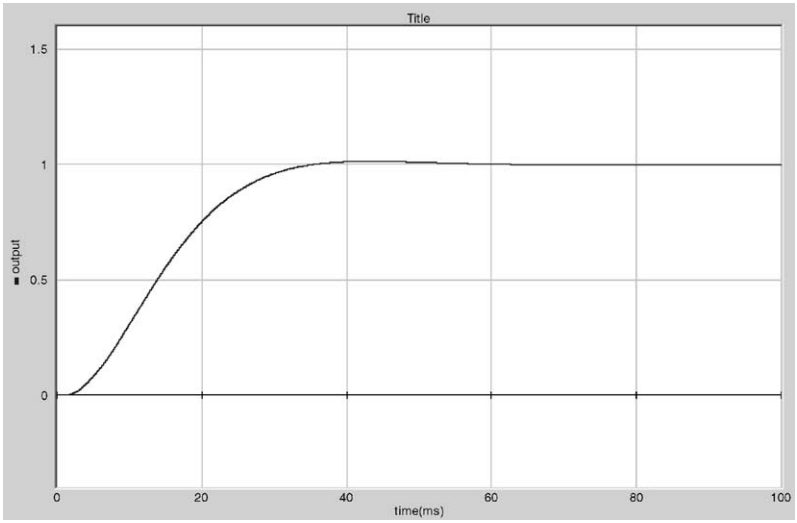

Fig. 39. Simulation result of evolved BG model 2.

embryo model (Fig. 35). The performance of this model is shown in Fig. 37. The position response for step function input quickly converges in 70 ms, which was an

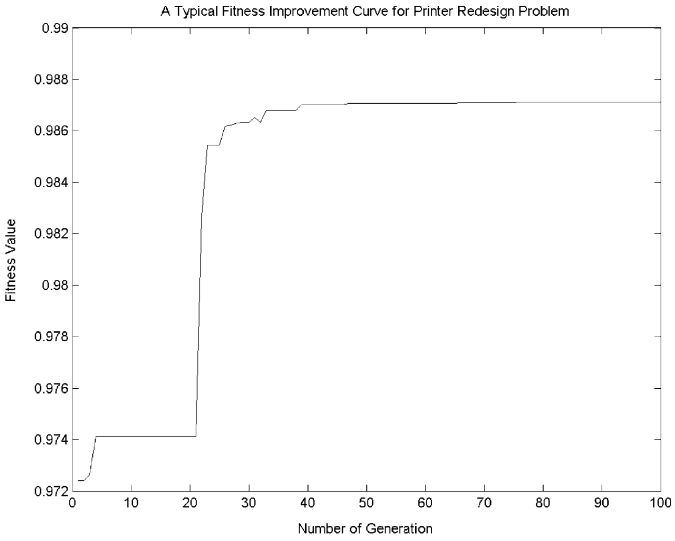

Fig. 40. Fitness history for a typical printer drive redesign run.

Table 6
Summary results of fitness for printer

	Fitness of printer	
	Distance	Fitness
1	15.076	0.985148
2	15.818	0.984428
3	15.188	0.985039
4	16.720	0.983555
5	15.053	0.985170
6	14.085	0.986111
7	15.122	0.985103
8	15.502	0.984734
9	15.132	0.985094
10	15.881	0.984367
Best	14.085	0.986111
Worst	16.720	0.983555
Average	15.358	0.984875
SD	0.6903	0.000669

acceptable timeframe. Another design is shown in Fig. 38. Four elements, 0-junction with C, 1-junction with R are added to modifiable site 2 and one R is added modifiable site 3 as shown in Fig. 38. Fig. 39 displays the performance of this model.

Table 6 represents the statistical results of 10 runs for the printer drive. The fitness history of a typical run is shown in Fig. 40.

7. Conclusion

This paper has suggested a new design methodology for automatically synthesizing designs for multi-domain, lumped parameter dynamic systems with a unified tool. A careful combination of BGs and GP, including a multi-step evaluation procedure that greatly increases the efficiency of fitness assessment, appears to be an appropriate approach to development of a method for synthesis of complex multi-domain systems, such as mechatronic systems.

As a proof of concept for this approach, evolution of BGs for three different domain design problems—a specified-target-eigenvalues design, electric filter design, and printer drive design—was tested. Experiments showed that all three yielded satisfactory results in moderate times and computational expenses.

This provides some support for the conjecture that much more complex multi-domain systems with more detailed performance specifications can be automatically designed, given longer execution times and/or using inexpensive cluster computing facilities. Further study will aim at extension and refinement of the design methodology and application to design of more complex and inter-domain mechatronic systems.

Acknowledgement

The authors gratefully acknowledge the support of the National Science Foundation through grant DMI 0084934.

References

[1] Youcef-Toumi K. Modeling, design, and control integration: a necessary step in mechatronics. IEEE/ASME Trans Mechatron 1996;1(1):29–38.
[2] Karnopp DC, Rosenberg RC, Margolis DL. System dynamics, a unified approach. 3rd ed. John Wiley & Sons; 2000.
[3] Rosenberg RC, Whitesell J, Reid J. Extendable simulation software for dynamic systems. Simulation 1992;58(3):175–83.
[4] Rosenberg RC. Reflections on engineering systems and bond graphs. Trans ASME J Dyn Syst, Measure Control 1993;115:242–51.
[5] Rosenberg RC. The ENPORT user's manual. E. Lansing, MI: Rosencode Associate; 1996.
[6] Rosenberg RC, Hales MK, Minor M. Engineering icons for multidisciplinary systems. Proc ASME IMECE 1996;DSC-V.58:665–72.
[7] Sharpe JE, Bracewell RH. The use of bond graph reasoning for the design of interdisciplinary schemes. In: 1995 International Conference on Bond Graph Modeling and Simulation. 1995. pp. 116–21.
[8] Youcef-Toumi K, Glaviano YA, Anderson P. Automated zero dynamics derivation from bond graph models. In: 1999 International Conference on Bond Graph Modeling and Simulation. 1999. pp. 39–44.
[9] Redfield RC. Bond graphs in dynamic systems designs: concepts for a continuously variable transmission. In: 1999 International Conference on Bond Graph Modeling and Simulation. 1999. pp. 225–30.

[10] Tay E, Flowers W, Barrus J. Automated generation and analysis of dynamic system designs. Res Eng Des 1998;10:15–29.
[11] Koza JR. Genetic programming: on the programming of computers by means of natural selection. The MIT Press; 1992.
[12] Koza JR. Genetic programming II: automatic discovery of reusable programs. The MIT Press; 1994.
[13] Koza JR, Bennett FH, Andre D, Keane MA. Genetic programming III, Darwinian invention and problem solving. Morgan Kaufmann Publishers; 1999.
[14] Koza JR, Bennett FH, Andre D, Keane M, Dunlap A. Automated synthesis of analog electrical circuits by means of genetic programming. IEEE Trans Evolution Comput 1997;1(2):109–28.
[15] Holland JH. Adaptation in natural and artificial systems. University of Michigan Press; 1975.
[16] Goldberg D. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley; 1989.
[17] Seo K, Goodman ED, Rosenberg RC. First steps toward automated design of systems using bond graphs and genetic programming. In: Proceedings of Genetic and Evolutionary Computation Conference, San Francisco. 2001. p. 189 (1-page abstract) and poster.
[18] Fan Z, Hu J, Seo K, Goodman ED, Rosenberg RC, Zhang B. Bond graph representation and GP for automated analog filter design. In: Genetic and Evolutionary Computation Conference Late-Breaking Papers, San Francisco. 2001. pp. 81–6, 2001.
[19] Luke S. 1997, Strongly-typed, multithreaded C genetic programming kernel. Available from: http://www.cs.umd.edu/users/-seanl/gp/patched-gp/.
[20] Zonker D, Punch WF. lil-gp 1.1 User's manual. GARAGe, College of Engineering, Michigan State University, 1998.
[21] Hu J, Goodman ED. Hierarchical fair competition model for parallel evolutionary algorithms, CEC 2002, Honolulu, Hawaii, May, 2002.