Exploring Multiple Design Topologies Using Genetic Programming and Bond Graphs

Zhun Fan
Electrical and Computer Engineering
Michigan State University
East Lansing, MI 48824

Kisung Seo
Genetic Algorithms Research and Applications Group
Michigan State University
2857 W. Jolly Rd., Okemos, MI 48864

Ronald C. Rosenberg
Dept. of Mechanical Engineering
Michigan State University
East Lansing, MI 48824

Jianjun Hu
Computer Science and Engineering
Michigan State University
East Lansing, MI 48824

Erik D. Goodman
Genetic Algorithms Research and Applications Group
Michigan State University
2857 W. Jolly Rd., Okemos, MI 48864

Abstract

To realize design automation of dynamic systems, there are two major issues to be dealt with: open-topology generation of dynamic systems and simulation or analysis of those models. For the first issue, we exploit the strong topology exploration capability of genetic programming to create and evolve structures representing dynamic systems. With the help of ERCs (ephemeral random constants) in genetic programming, we can also evolve the sizing of dynamic system components along with the structures. The second issue, simulation and analysis of those system models, is made more complex when they represent mixed-energy-domain systems. We take advantage of bond graphs as a tool for multi- or mixed-domain modeling and simulation of dynamic systems. Because there are many considerations in dynamic system design that are not completely captured by a bond graph, we would like to generate multiple solutions, allowing the designer more latitude in choosing a model to implement. The approach in this paper is capable of providing a variety of design choices to the designer for further analysis, comparison and trade-off. The approach is shown to be efficient and effective in an example of open-ended real-world dynamic system design application, a printer re-design problem.

1 INTRODUCTION

In general, design of dynamic systems includes two steps: conceptual design and detailed design. In the conceptual design phase, the following questions should be answered (Tay *et al.* 1998): 1) What is the exact design problem to be solved? (This requires a complete and consistent listing of the requirements), and 2) what are the key problem areas in the solution? (This requires the identification of critical parts of the solution that will determine the performance). Then the process of detailed design can be undertaken, identifying those candidate solutions that meet the requirements and provide the level of performance needed. The research in this paper focuses on the detailed design of dynamic systems. The strategy is to develop an automated procedure capable of exploring the search space of candidate dynamical systems and providing design variants that meet desired design specifications or dynamical characteristics. The method must be able to explore the design space in a topologically open-ended manner, yet still find appropriate configurations efficiently enough to be useful.

Much research has been done on design automation of single domain systems using an evolutionary computation approach. For example, automated design of analog circuits has attracted much attention in recent years (Grimbleby, 1995) (Lohn, 1999) (Koza, 1999) (Zhun, 2000). It could be classified into two categories: GA-based and GP-based. Most GA-based approaches realize topology optimization via a GA and parameter optimization with numerical optimization methods (Grimbleby, 1995). Some GA approaches also evolve both topology and component parameters; however, they typically allow only a limited number of components to be evolved (Lohn, 1999). Although their work basically achieves good results in analog circuit design, it is not easily extendable to interdisciplinary systems like mechatronic systems.

Design of interdisciplinary (multi-domain) dynamic engineering systems, such as mechatronic systems, differs from design of single-domain systems, such as electronic circuits, mechanisms, and fluid power systems, in part because of the need to integrate the several distinct domain characteristics in predicting system behavior (Coelingh *et al.*). However, most current modeling and simulation tools that provide for representation at a schematic, or topological, level have been optimized for a single domain. The bond graph provides a unified model representation across inter-disciplinary system domains. Tay uses bond graphs and GA to generate and analyze dynamic system designs automatically (Tay *et al.* 1998). He uses nested GA to evolve both topology and parameters for dynamic systems. However, the efficiency of his approach is hampered by the weak ability of GA to search in both topology and parameter spaces simultaneously.

Genetic programming is an effective way to generate design candidates in an open-ended, but statistically structured, manner. There have been a number of research efforts aimed at exploring the combination of genetic programming with physical modeling to find good engineering designs. Perhaps most notable is the work of Koza *et al.*. He presents a single uniform approach using genetic programming 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 approach appears to be very promising, having produced a number of patentable designs for useful artifacts. It is closely related to our approach, except that it searches in a single energy domain.

We investigate an approach combining genetic programming and bond graphs to automate the process of design of dynamic systems to a significant degree. To improve the topology search capability of GP and enable it to provide a diversity of choices to the designer, a special form of parallel GP, the Hierarchical Fair Competition GP (HFC-GP), is used in this paper (Hu, *et al.*, 2002). The efficiency and effectiveness of the approach are illustrated in an interesting redesign example involving the drive mechanism for an electric printer. Several design alternatives for the printer drive are derived through exploring open-topologies in bond graph space. It turns out that some of them are obviously physically realizable and others are not.

2 DESIGN DOMAIN AND METHODOLOGY

2.1 MULTI-DOMAIN DYNAMIC SYSTEMS

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 (Figure 1 a)) while "RLC" electric circuits include resistors, inductors and capacitors (Figure 1 b)). However, they could both be expressed in the same bond graph (Figure 1 c)).

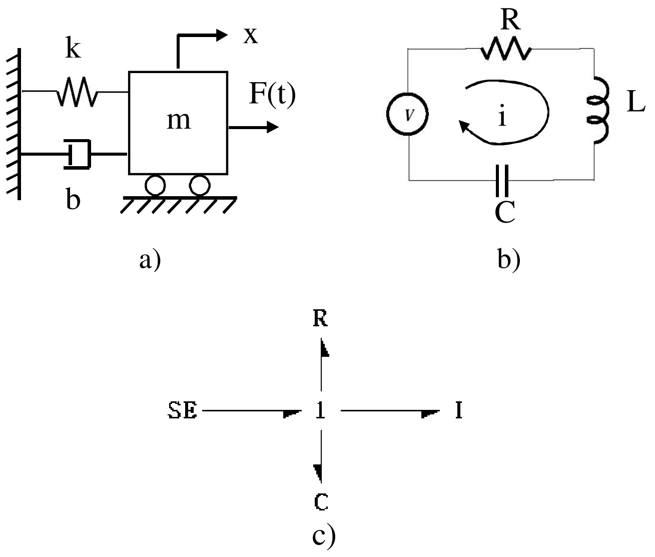


Figure 1. Dynamic systems and bond graph representation : a) mechanical, b) electrical , and c) bond graph that represents both

2.2 BOND GRAPHS

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems, especially hybrid multi-domain systems including mechanical, electrical, pneumatic, hydraulic, etc. (Karnopp *et al.* 2000). It is the explicit representation of model topology that makes the bond graph a good candidate for use in open-ended design searching. For notation details and methods of system analysis related to the bond graph representation see Karnopp *et al.* and Rosenberg (Rosenberg *et al,* 1992). Much recent research has explored the bond graph as a tool for design (Sharpe and Bracewell 1995, Tay *et al.* 1998, Youcef-Toumi 1999, Redfield 1999).

In our research, the bond graph has additional desirable characteristics for selection as the tool for system representation and simulation. The evaluation efficiency of the bond graph model can be improved because analysis of causal relationships and power flow between elements and subsystems can be done quickly and easily, and reveals certain important 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. Because virtually all of the circuit topologies passing causal analysis can be simulated, 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.

Another characteristic of bond graphs is their ease of mapping to the engineering design process (Xia, *et al.* 1991). Because each component of the system can be represented correspondingly in a bond graph, junctions and elements can be added to or deleted from a model without causing dramatic changes. This emulates the engineering process of modifying systems, refining simple designs discovered initially, adding size and complexity as needed to meet more complicated design demands step by step. As genetic programming usually shows a weak causality of structure evolution (Rosca, 1995), this potential strong causality of the bond graph modification process also makes bond graph representation an attractive technique to use in genetic programming to explore the open-ended dynamic system design space in an evolutionary process.

2.2 GENETIC PROGRAMMING AND BOND GRAPHS

The tree representation on GP chromosomes, as compared with the string representation typically used in GA, gives GP more flexibility to encode solution representations for many real-world design applications. The bond graph, which can contain cycles, is not represented directly on the GP tree—instead, the function set (nodes of the tree) encode a constructor for a bond graph.

We define the GP functions and terminals for bond graph 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). Details of function definitions are illustrated in Seo *et al*. (2001).

Table 1 Function and terminal set for bond graph evolution

Name	Description
add_C	Add a C element to junctions
add_I	Add an I element to junctions
add_R	Add an R element to junctions
insert_J0	Insert a 0-junction in bond
insert_J1	Insert a 1-junction in bond
replace_C	Replace current element with C element
replace_ I	Replace current element with I element
replace_ R	Replace current element with R element
+	Add two ERCs
-	Subtract two ERCs
endn	End terminal for add element operation
endb	End terminal for insert junction operation
endr	End terminal for replace element operation
erc	Ephemeral random constant (ERC)

2.3 DESIGN PROCEDURE

The flow of the entire algorithm is shown in Figure 2. The user specifies 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) After that, an initial population of GP trees is randomly generated. Each GP tree maps to a bond graph tree. Analysis is then performed on each bond graph tree. This analysis consists of two steps – causal analysis and state equation analysis. After the (vector) state equation is obtained, the important dynamic characteristics of the system are sent to the fitness evaluation module and the fitness of the tree is evaluated. For each evaluated and sorted population, genetic operations – selection, crossover, mutation and reproduction – are carried out to seek design candidates with improved quality. The loop of bond graph analysis and GP operation is iterated until a termination condition is satisfied or a specified number of iterations performed. The final step is to instantiate a physical design, replacing the bond graph with the physical components represented.

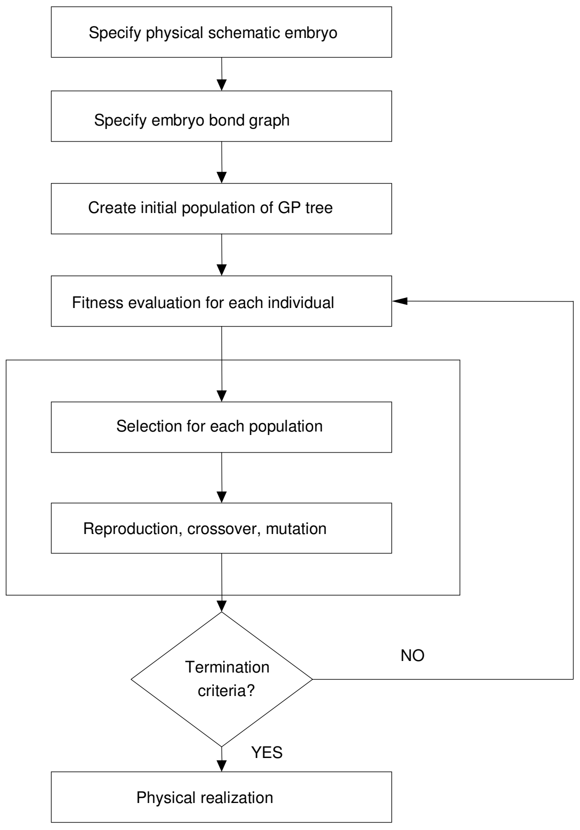

Specify physical schematic embryo
Specify embryo bond graph
Create initial population of GP tree
Fitness evaluation for each individual
Selection for each population
Reproduction, crossover, mutation
Termination criteria?
NO
YES
Physical realization

Figure 2. Flow chart of the design procedure

3 CASE STUDY

3.1 PROBLEM FORMULATION

The original problem was presented by C. Denny and W. Oates of IBM, Lexington, KY, in 1972. Figure 3 shows a closed-loop control system to position a rotational load (inertia) denoted as J_L. The problem with the design is that the position output of the load J_L has intense vibrations (see Figure 4). The design specification is to reduce the vibration of the load to an acceptable level, given certain command conditions for rotational position.

We want the settling time to be less than 70ms when the input voltage is stepped from zero to one. Note that the settling time of the original system is about 2000ms. The time scale in Figure 4 is 4000 ms.

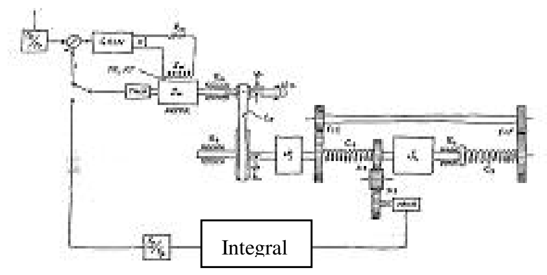


Figure 3. Schematic of the printer drive system

The system includes electric voltage source, motor and mechanical parts. As it is a multi-domain system, a bond graph is convenient to use for modeling.

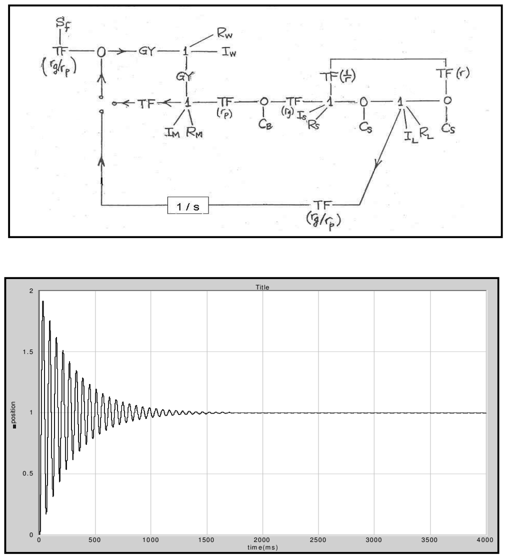


Figure 4. a) Bond graph model b) Positional vibration of the load

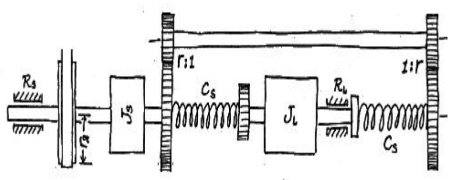


Figure 5. The embryo subsystem

By analyzing the model, we conclude that the critical part for the design is a subsystem that involves the drive shaft and the load (Figure 5). The input is the driving torque, T_d, generated through the belt coupling back to the motor (not shown).

This subsystem was deemed a logical place to begin the design problem. The questions left to the designer now are: 1) at which exact spots of the subsystem new components should be inserted, 2) which types of components and how many of them should be inserted, in which manner, and 3) what should be the values of the parameters for the components to be added? The approach reported in this paper is able to answer these three questions in one stroke in an automated manner, once the embryo system has been defined.

3.2 AN EMBRYO FOR EVOLUTION

To search for a new design using the BG/GP design tool, an embryo model is required. The embryo model is the fixed part of the system and the starting point for GP to generate candidates of system designs by adding new components in a developmental manner. The embryo used for this example, expressed in bond graph language, is shown in Figure 6, with the modifiable sites highlighted. The modifiable sites are places that new components can be added. The choice of modifiable sites is typically easy for the designer to decide. However, modifiable sites are only *possible* spots for insertion of new components – they are not necessarily inserted to any particular one of them. In this particular example, designers need have no idea whether assemblies of new components will be inserted at modifiable site (1), or at modifiable site (2) , at site(3), or at any combinations of them. Instead, the algorithm will answer these questions in an automatic way, without intervention by the human designer.

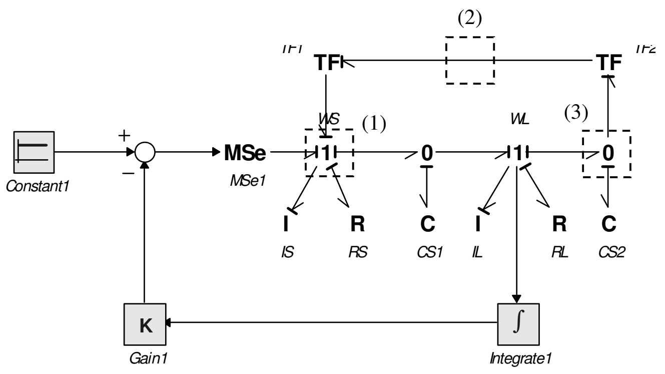


Figure 6. Bond graph model for the embryo system

The parameters for the embryo model are:

$I_s: 6.7\times10^{-6}\, kg\cdot m^2$

$R_s: 0.013\times10^{-3}\, N\cdot m\cdot \sec \big/ rad$

$C_{s1}:\ 0.208\quad N\cdot m\cdot \big/ rad$

$C_{s2}:\ 0.208\quad N\cdot m\cdot \big/ rad$

R_L: $0.58 \times 10^{-3}\, N \cdot m \cdot \sec / rad$

I_L: $84.3 \times 10^{-6}\, kg \cdot m^2$

3.3 THE HFC MODEL OF PARALLEL GENETIC PROGRAMMING

A special form of parallel GP, HFC-GP is applied in this research. In the HFC (Hierarchical Fair Competing) model (Fig 7), multiple subpopulations are organized in a hierarchy, in which each subpopulation can only accommodate individuals within a specified range of fitnesses (Hu *et al*, 2002). New individuals are created continuously in the bottom layer. Use of the HFC model balances exploration and exploitation of GP effectively. Our experience shows that using the HFC model can also substantially increase the topological diversity of the whole population and help to provide the designer a diverse set of competing design candidates for further trade-offs.

3.4 DEFINITION OF FITNESS FUNCTION

The fitness function of individual design is defined

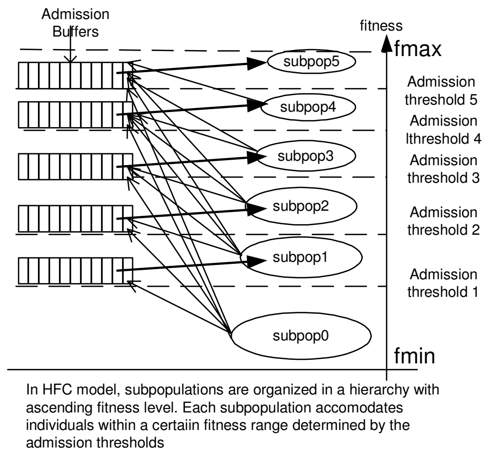

Admission Buffers
fitness
fmax
subpop5
subpop4
subpop3
subpop2
subpop1
subpop0
Admission threshold 5
Admission lthreshold 4
Admission threshold 3
Admission threshold 2
Admission threshold 1
fmin
In HFC model, subpopulations are organized in a hierarchy with ascending fitness level. Each subpopulation accomodates individuals within a certaiin fitness range determined by the admission thresholds

Figure 7. Hierarchical Fair Competition Model in GP

according to the position output response of the load J_L as follows.

Within the time range of interest (0~500ms in this example), uniformly sample 1000 points of the output response (yielding a time interval between two adjacent sampling points of 0.5ms). Compare the magnitudes of the position output of the load J_L at the sample points with target magnitudes (unity in this example), compute their difference and get a squared sum of differences as raw fitness, defined as $Fitness_{raw}$. Then calculate normalized fitness according to:

$$Fitness_{norm} = 0.5 + 1000 \;/\left(2000 + Fitness_{raw}\right)$$

It can be assumed approximately that the higher the normalized fitness, the better the design. Two reasons make the fitness definition an approximate one: 1) it does not reflect directly the strict definition of settling time, and 2) it does not include other considerations in design of the system except output response. A modified fitness function could be defined later if required. However, in this research, the definition is enough to manifest the feasibility and efficiency of the approach reported. The design results achieved (Figures 9-16) show performances satisfying the design specification presented in this research.

3.5 EXPERIMENTAL SETUP

We used a strongly-typed version [Luke, 1997] of lilgp [Zongker and Punch, 1996] to generate bond graph models. The major GP parameters were as shown below:

Number of generations:	500
Population size:	500
Initial population:	half_and_half
Initial depth:	4-6
Max depth:	16
Max nodes:	1000
Selection:	tournament (size=7)
Crossover:	0.8
Mutation:	0.2

Three major code modules were created in our work. The algorithm kernel of HFC-GP was a modified version of an open software package developed in our research group -- lilgp. A bond graph class was implemented in C++. The fitness evaluation package is C++ code converted from Matlab code, with hand-coded functions used to interface with the other modules of the project. The commercial bond graph software package 20Sim was used to verify the dynamic characteristics of the evolved design.

The GP program obtains satisfactory results on a Pentium-IV 1GHz in 5~15 minutes, which shows the efficiency of our approach in finding good design candidates.

3.6 EXPERIMENTAL OBSERVATIONS

The fitness improvement curve of GP algorithm is plotted versus generation number in Figure 8.

Three competing design candidates with different topologies, as well as their performances and possible physical realizations, are provided in Figures 9-16. We can see from the output rotational position responses that they all satisfy the design specification of settling time less than 70ms. Note that the time scale of the plots is 100 ms.

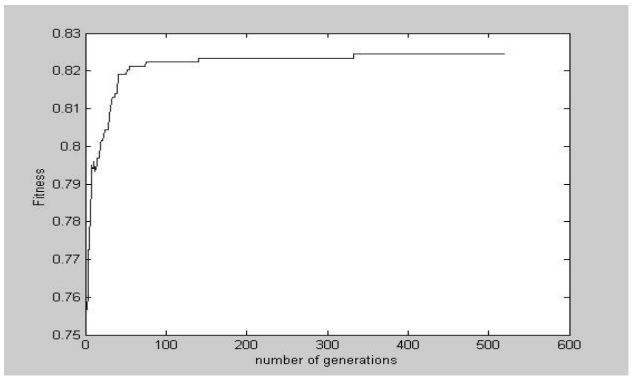
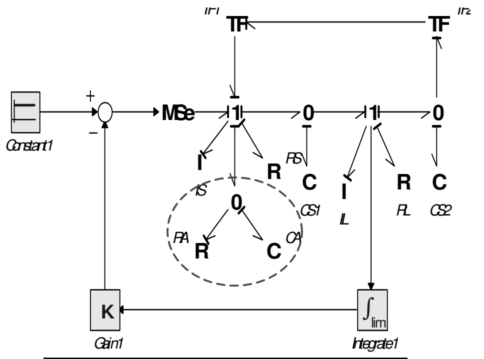


Figure 8. Fitness Improvement Curve

Design variant 1 is represented in Figure 9. Two new components (R, C) are added with a 0-junction at modifiable site (1). The parameters of the components added are also given. Dashed circles highlight the newly evolved components in the bond graph figures. Figure 10 displays rotational position output for variant 1.



RA: 12.6957E-03 N m sec / rad
CA: 0.1962 N m / rad

Figure 9. Bond graph model of design variant 1

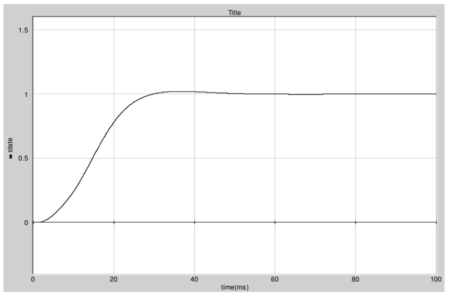


Figure 10. Position output of design variant 1

Figure 11 gives one possible physical realization of design variant 1. One damper and one spring are connected to the embryo model.

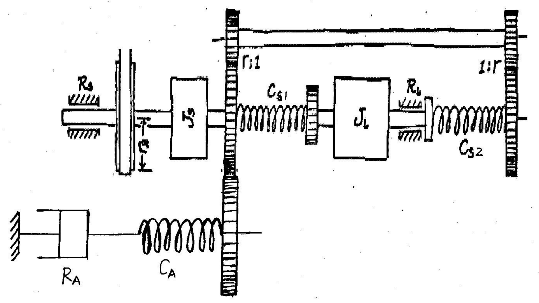
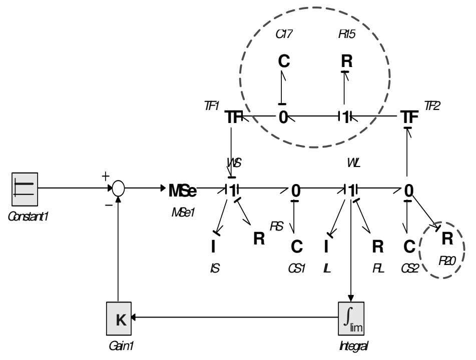


Figure 11. Physical realization of design variant 1

Variant 2 adds new components to modifiable site (2) and modifiable site (3) as shown in Figure 12. Figure 13 displays rotational position output for variant 2.



R20: 75.101E-03 N m sec / rad	
R15: 0.142E-03 N m sec / rad	C17: 10.000 N m / rad

Figure 12. Bond graph model of design variant 2

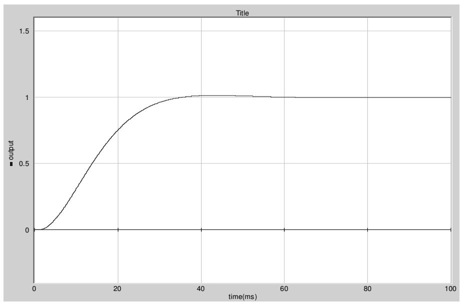


Figure 13. Position output of design variant 2

Figure 14 gives one possible physical realization of design variant 2.

Design variant 3 is represented in Figure 15. Variant 3 adds new components to modifiable site (1) and modifiable site (2). Figure 16 displays rotational position output for variant 3. The bond graph model of variant 3 is not obviously physically realizable, because component I is attached to a 0-junction.

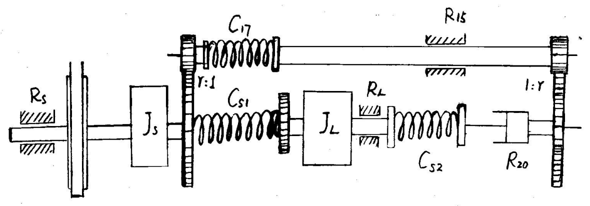
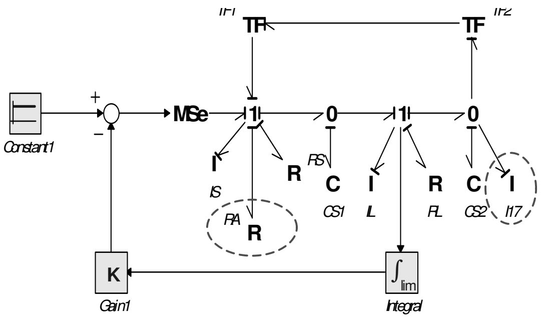


Figure 14. Physical realization of design variant 2



RA: 13.899E-03 N m sec / rad

I17: 15.022E-06 kg m^2

Figure 15. Bond graph model of design variant 3

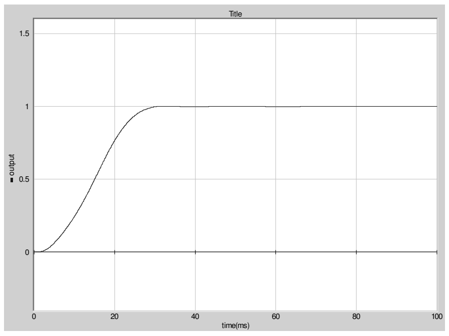


Figure 16. Position output of design variant 3

It is clear that the approach reported in this research is both efficient and effective, capable of providing designers with a variety of design alternatives. This gives designers considerable flexibility to generate and to compare a large variety of design schemes.

4 CONCLUSIONS

This research has explored a new automated approach for synthesizing designs for dynamic systems. By taking advantage of genetic programming as a search method for competent designs and the bond graph as a representation for dynamic systems, we have created a design environment in which open-ended topological search can be accomplished in an automated and efficient manner and the design process thereby facilitated.

The paper illustrates the process of using this approach in detail through a printer redesign problem. Bond graphs have proven to be an effective tool for both modeling and design in this problem. A special form of parallel GP, the Hierarchical Fair Competition-GP, has been shown to be capable of providing a diversity of competing designs with great efficiency.

We plan to continue our research by identifying improved methods for minimizing the occurrence of unrealizable bond graph designs. One such approach is to use multi-objective evolutionary computation to shape the results. A second approach is to use a set of revised GP operators to build bond graphs that avoid unrealizable models.

Acknowledgments

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

References

H. J. Coelingh, T. de Vries, and J. Amerongen (1998) Automated Performance Assessment of Mechatronic Motion Systems during the Conceptual Design Stage. *Proc. 3rd Int'l Conf. on Adv. Mechatronics*, Okayama, Japan.

Z. Fan, J. Hu, K. Seo, E. Goodman, R. Rosenberg, and B. Zhang (2001). Bond Graph Representation and GP for Automated Analog Filter Design, *Proceedings of the Genetic and Evolutionary Computation Conference*: 81-86.

J. B. Grimbleby (2000). Automatic analogue circuit synthesis using genetic algoriths. *IEE Proc. – Circuits Devices Syst.* : 319-323.

J. Hu, E. D. Goodman (2002). The Hierarchical Fair Competition (HFC) Model for Parallel Evolutionary Algorithms. *Congress on Evolutionary Computation* .

D. C. Karnopp, D. L. Margolis and R. C. Rosenberg (2000). *System Dynamics: Modelling and Simulation of Mechatronic Systems. Third Edition.* New York: John Wiley & Sons, Inc.

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

J. R. Koza, F. H. Bennett III, D. Andre and M. A. Keane (1999b). The design of analogue circuits by means of genetic programming. In P. J. Bentley (ed.), *Evolutionary Design by Computers*, 365-385. London: John Wiley & Sons Ltd.

J. D. Lohn, S. P. Colombano (1999). A circuit representation techniques for automated circuit design. *IEEE Transactions on Evolutionary Computation*: 205-219.

S. Luke, 1997, *Strongly-Typed, Multithreaded C Genetic Programming Kernel,* http://www.cs.umd.edu/users/-seanl/gp/patched-gp/.

H. M. Paynter (1991). An epistemic prehistory of bond graphs. In P. C. Breedveld and G. Dauphin-Tanguy (ed.), *Bond Graphs for Engineers,* 3-17. Amsterdam, The Netherlands: Elsevier Science Publishers.

R.C. Redfield (1999). Bond Graphs in Dynamic Systems Designs: Concepts for a Continuously Variable Transmission. *International Conference on Bond Graph Modeling and Simulation*: 225-230.

J. P. Rosca , D. H. Ballard (1995), Causality in genetic programming. In L. Eshelman (ed.), Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95), 256-263. San Francisco, CA: Morgan Kaufmann.

R.C. Rosenberg, J. Whitesell, and J. Reid (1992). Extendable Simulation Software for Dynamic Systems. *Simulation* 58: 175-183.

K. Seo, E. D. Goodman and R. C. Rosenberg (2001). First steps toward automated design of mechatronic systems using bond graphs and genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference* : 189.

J.E. Sharpe , and R.H. Bracewell (1995). The Use of Bond Graph Reasoning for the Design of Interdisciplinary Schemes. *International Conference on Bond Graph Modeling and Simulation:* 116-121.

E. H. Tay, Woodie Flowers and John Barrus (1998). Automated Genration and Analysis of Dynamic System Designs. *Research in Engineering Design* 10: 15-29.

S. Xia, D. A. Linkens and S. Bennett (1991). Integration of qualitative reasoning and bond graphs: an engineering approach. In P. C. Breedveld and G. Dauphin-Tanguy(ed.), *Bond Graphs for Engineers,* 323-332. Amsterdam, The Netherlands: Elsevier Science Publishers.

K. Youcef-Toumi, Y. Ye., A. Glaviano, and P. Anderson (1999). Automated Zero Dynamics: Derivation from Bond Graph Models. *International Conference on Bond Graph Modeling and Simulation:* 39-44.

D. Zongker and W.F. Punch, III, 1998, *lil-gp 1.1 User's Manual*, GARAGe, College of Engineering, Michigan State University.