Automated Design of Mechatronic Systems: Novel Search Methods and Modular Primitives to Enable Real-World Applications

Erik D. Goodman
Michigan State University
238 PUBLICATIONS 3,946 CITATIONS
SEE PROFILE

Zhun Fan
Shantou University
151 PUBLICATIONS 999 CITATIONS
SEE PROFILE

Jianjun Hu
China University of Mining Technology
93 PUBLICATIONS 887 CITATIONS
SEE PROFILE

Ronald C. Rosenberg
Michigan State University
82 PUBLICATIONS 2,708 CITATIONS
SEE PROFILE

Principal Investigator/Project Director: Erik D. Goodman

Institution: Michigan State University

Award Number: DMI-0084934 Program: Engineering Design

Project Title: Automated Design Methodology for Mechatronic Systems Using Bond Graphs and Genetic Programming

Automated Design of Mechatronic Systems: Novel Search Methods and Modular Primitives to Enable Real-World Applications

Erik D. Goodman*+, Kisung Seo*+, Zhun Fan*+, Jianjun Hu*#, Ronald C. Rosenberg++
{ goodman, ksseo, fanzhun, hujianju, rosenber@egr.msu.edu }

***Genetic Algorithms Research and Applications Group (GARAGe)**
+Department of Electrical and Computer Engineering
#Department of Computer Science and Engineering
++Department of Mechanical Engineering
Michigan State University
East Lansing, MI 48824-1226

Abstract

This paper suggests not only an automated design methodology for synthesizing designs for multi-domain systems, such as mechatronic systems, but also a type of fine control of modularity and several innovative GP search methods. The domain of mechatronic systems includes mixtures of, for example, electrical, mechanical, hydraulic, pneumatic, and thermal components, making it difficult to design a system to meet specified performance goals using a single design tool. 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 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 bond graphs) with ever-improving performance, in an iterative loop of synthesis, analysis, and feedback to the synthesis process.

We are reporting advances in methods for automated design of two types: use of new, more complex design primitives and two improvements to the genetic programming algorithms used in the design synthesis: SFS (Structure Fitness Sharing) and HFC (Hierarchical Fair Competition). The improvements have been developed and tested during investigations of four 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 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. The fourth is in the hydraulic domain – design of an air pump to maximize outflow subject to a constraint specifying maximum power consumption.

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 parts 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 (Youcef-Toumi [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.

Bond graphs (Karnopp *et al.*[2], Rosenberg [3-6]) 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. Sharpe and Bracewell [7] present the use of bond graph 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 bond-graph-inspired rules. Youcef-Toumi [8] introduces an algorithm that identifies automatically the physical components and/or subsystems that are responsible for zero dynamics. Redfield [9] demonstrates the value of using bond graphs 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 bond graph models. This approach adopts a variational design method, which means they make a complete bond graph model first, then change the bond graph 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.

Genetic programming 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 genetic programming with physical modeling to find good engineering designs. Perhaps most notable is the work of Koza *et al.* [11-13]. 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. 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.

Based on the novel design methodology we have already developed to combine bond graphs and genetic programming, we have recently added several key features that make this approach more effective.

The first is the introduction of specially-structured, larger modules for use as primitives in the GP/bond graph search, Such modules, comprising assemblies of the primitive bond graph elements we have earlier used, continue to span the entire design space covered by the basic primitives, with the exception of some topologies that would later be found causally unacceptable for implementation or capable of functional simplification to an equivalent-performance structure. Thus, the advanced, modular primitives gain by eliminating many useless regions of the design space, thus speeding search and making it more robust, without any negative impacts on the quality of the designs that can be produced. Our observations of the designs emerging from our test problems suggested the appropriateness of modules of the level now used as the primitives; in our continuing work, we are looking for additional modules at yet higher levels that may become apparent as useful elements to speed search without compromising its capability to span the design space.

In parallel with the use of higher-level modular primitives, addition of two novel features to our GP search algorithms has also resulted in order-of-magnitude reductions in typical search times to good designs. Both techniques help to address the tendency of GP to converge prematurely on designs that are only locally optimal, rather than to search at greater length for globally optimal solutions. The first of these methods, *Structure Fitness Sharing* ("SFS"), is used to provide better control of the tradeoff between search for new design topologies (component types and their interconnections) versus selection of appropriate values for their parameters (i.e., mass, capacitance, spring constant, etc.) The second, *Hierarchical Fair Competition* ("HFC"), can be used in many

forms of evolutionary competition, and facilitates the preservation of diversity in the search population while simultaneously allowing the exploitation of good "building blocks" discovered during the search process.
Through a series of experiments, we have shown that the performance of our system for automated design is greatly improved by use of these novel techniques.

This paper illustrates the effectiveness of our design methodology for applications in each of several domains. Section 2 discusses the inter-domain nature, efficient evaluation and graphical generation of bond graphs. Section 3 describes evolution of bond graphs by genetic programming. Section 4 describes the two novel search enhancements, SFS and HFC. Sections 5-8 present some results for four problems: design for eigenvalue placement, analog filter design, printer drive redesign, and air pump redesign, and Section 9 concludes the paper.

2. A Design Methodology Based on Bond Graphs and GP

2.1 Unified and Automated Methodology for Multi-Domain System Design

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, etc. Designers sometimes have to consume large amounts of time to prepare new analysis tools or methods. A design methodology that combines bond graphs and genetic programming can provide both capabilities – an automated and unified approach (Figure 2.1). 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 genetic programming 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), even the simplest lumped-parameter dynamical mechanical systems models typically include at least masses, springs and dampers (Figure 2.2a) while "RLC" electric circuits include resistors, inductors and capacitors (Figure 2.2b). Figure 2.3 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. Figure 2.2 shows examples of single-domain systems, while Figure 2.3 represents a mixed-domain system.

2.2 Background: Bond Graphs and Genetic Programming

Topologically, bond graphs consist of *elements* and *bonds*. Relatively simple systems include passive one-port elements C, I, and R, active one-port elements S_e and S_f, and two-port elements TF and GY (transformers and gyrators). These elements can be attached to *0- (or 1-) junctions*, which are multi-port elements, using bonds. Figure 2.4 consists of S_e, 1-junction, C, I, and R elements, and that same bond graph represents, for example, either a mechanical mass, spring and damper system (Figure 2.2a), or an RLC electric circuit (Figure 2.2b). S_e corresponds with force in mechanical systems, or voltage in electrical. 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 mechanical system, or implies that the current in the RLC loop is common. The R, I, and C represent the damper, inertia (of a 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 Figure 2.5. 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, the TF corresponds to a gear ratio or signal ratio, while the GY relates gain to voltage or current in a motor to mechanical rotation.

Genetic programming is an extension of the genetic algorithm, using evolution to optimize actual computer programs or algorithms to solve some task (Holland [15], Goldberg [16]), typically involving a graph-type (or other variable-length) representation. The most common form of genetic programming is due to John Koza [11-13], and uses trees to represent the entities to be evolved. Genetic programming can manipulate variable-sized strings and can be used to "grow" trees that specify increasingly complex bond graph models, as described below.

If the scope and analysis efficiency of the bond graph model can be successfully integrated with the impressive search capability of genetic programming when utilized to its full potential, an extremely capable automated synthesis procedure, without need for user intervention, should result.

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, bond graphs can contain loops, so we do not represent the bond graphs directly as our GP "chromosomes." Instead, a GP tree specifies a *construction procedure* for a bond graph. Bond graphs are "grown" by executing the sequence of GP functions specified by the tree, using the bond graph embryo as the starting point.

Initial studies were reported in Seo *et al.*[17] and Fan *et al.*[18]. The following set of bond graph 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 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 .

3.2 Modular Primitives

In our previous experiments with the Bond Graph/Genetic Programming design methodology, we observe many instances of redundant or unnecessary junctions and elements that could be removed without altering of any physical meanings. To reduce this redundancy, a new design for the GP function sets has been devised. First, the concept of a *regular* junction set is utilized, which avoids introduction of adjacent pairs of 0-junctions, or pairs of 1-junctions, in the bond graph (desirable because such adjacent pairs can be coalesced into a single 0- or 1-junction with identical physical meaning). Experiments (not included here), show that elimination of these redundancies does not hinder at all the overall performance, but produces more compact designs.

Second, we have introduced a set of causally well-posed primitives that generate only designs satisfying the causality principles formerly used to "screen out" causally ill-formed designs. We did this introduction in two steps – first, one that also constrains somewhat the search space, and second, an improved system that eliminates these constraints. This first set of primitives consists of 0- (or 1-) junctions to which are pre-attached C, I, and R elements (one of each) (Figure 3.1a). Because all designs generated are now causally feasible, the need for the prior causality check prior to evaluation of the dynamic system properties is eliminated. This approach has another constraint -- that the embryo should be also satisfy the causality conditions. This approach was tested on the eigenvalue design problem, and worked very well. It yielded better performance with much less computation time. Table 3.2 shows the comparison results for the eigenvalue placement problem (see Section 5) between the *Basic set* and the *Causally well-posed set*. However, it cannot generate all possible design topologies, except by painstakingly eliminating components by setting their parameter values to zero or infinity (as appropriate), thereby effectively eliminating them. Such a set may not work well at all for other applications. However, a modified form (described next) eliminates this difficulty and yields even better performance.

The *Complex* (or Combined) set of primitives was developed, which allows the free combination of C, I, and R elements attached to 0- (or 1-) junctions using on/off switches to control the presence/absence of each C, I, and R element (see Figure 3.1b). This set of primitives does not introduce any constraints on possible design topologies, but retains significant advantages over the basic set. It is more modular than the basic set and more flexible than the causally well-posed set. This new set of primitives has only been implemented recently, so not many experiments have been run; however, performance to date appears very promising.

If this approach can be combined with careful control of search (as discussed below) it promises very good solutions in both performance and computation time. This modular construction set facilitates flexible specification of middle-level modularity, which can yield improvements in high-level functionality and simultaneous decreases in computational effort. For example, the system can control the relative frequencies at which mutation operations change a junction with C and R elements turned on to a node with I and R elements turned on – as opposed to one with only the C element turned on, for example. Studies in particular design domains may increase our understanding, first, of empirical principles yielding increased design quality and efficiency, and, later, of design concepts that make the effectiveness of such second-level skewing of probabilities "obvious."

3.3 Overall Design Procedure

The flow of the entire algorithm is as follows. 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 bond graph model and corresponding embryo (starting) element for a GP tree. From that, an initial population of GP trees is randomly generated. Bond graph analysis is then performed on the bond graph specified by each tree. This analysis consists of two steps – causal analysis (unless the new procedures are used that guarantee satisfaction of causality constraints) and then, 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 bond graph 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 bond graph in physical components, which is partially done (for the analog filter) in the current work. As mentioned earlier, a two-stage evaluation procedure is executed to evaluate bond graph models. The first, causal analysis (Karnopp *et al.*[2]), allows rapid determination of feasibility of candidate designs, thereby sharply reducing the time needed for analysis of designs that are infeasible. For those designs "passing" the causal analysis, the state model is automatically formulated.

4. Enhanced Search Algorithms to Overcome Local Minima

4.1 Structure Fitness Sharing (SFS)

Genetic programming with a developmental approach has been demonstrated to be exceptionally useful for open-ended system structure innovation for many applications. However, in all of these problems, GP often suffers severely from the premature convergence of structures, such that some structures with relatively good parameters tend to dominate the population and then prevent other promising structures from being found.

While fitness sharing has been widely used in evolutionary computation to address premature convergence, it is normally applied based on *clustering* of individuals using a distance metric. DeJong *et al.* [19] use the multi-objective method to explicitly promote diversity by adding a diversity objective. In their method, a distance measure defined as follows is used in the diversity objective. The distance between two trees is the sum of the distances of the corresponding nodes – i.e., distances between nodes that overlap when the two trees are overlaid, starting from the root. An improved version of the above distance metric between two trees is proposed in Ekart and Nemeth [20] and used to do fitness sharing in GP. The computational cost of this approach is still demanding for a complex structure and its parameters often require a big tree. Also but more importantly, the underlying assumption of the above distance metrics is that structural dissimilarity measured between two GP trees meaningfully reflects the dissimilarity in function between the two structures. However, as the structure space represented by a GP tree is a highly non-linear space, in most cases, a change of a single (non-parameter) node changes the behavior of the GP tree dramatically.

Therefore, we provided a new approach, SFS (Structure Fitness Sharing) to maintain the structural diversity of a population, based on a structure labeling technique. SFS achieves balanced structure and parameter search by applying fitness sharing to each unique structure in a population, to prevent takeover by the best structure and thereby maintain the diversity of both structures and parameters simultaneously. The difference in SFS is that the distance is considered to be infinite if two structures are different, and zero if the structures are identical (but with possibly different parameters). The recognition in SFS that a distance metric among individuals with different structures is typically rather arbitrary has important benefits in terms of both efficiency of fitness calculation and globality of search.

In contrast to the GA fitness sharing using a distance measure to identify peaks, in SFS, fitness sharing is based on the tree structures, treating each tree structure in GP as a peak in the space of parameters and structures. SFS works by penalizing those structures which have too many individuals (i.e., same structure but different parameter values) in the current population. With this method, each structure has a chance to have its parameter space searched. Premature convergence of structures is limited, and we can still devote more effort to high-fitness structure search. The SFS algorithm is described as follows.

The effectiveness of SFS was demonstrated on a real-world bond-graph-based analog circuit synthesis problem, called here the eigenvalue placement problem – namely, to evolve a circuit that has a pre-specified set of eigenvalues. We applied SFS to 6-, 8-, and 10-eigenvalue problems and compared it to our standard single-population and multi-population GP methods for the same problems. The experimental results show that SFS achieves much better results. Details are provided in Hu *et al.*[22].

4.2 Hierarchical Fair Competition (HFC) Model for Parallel Evolutionary Algorithms

Inspired by the fair competition principle often seen in society and biology, HFC is designed to ensure fair competition in evolutionary algorithms by stratifying subpopulations into different fitness levels (Figure 4.1), in a manner similar, for example, to the admission mechanism in educational systems of many developing countries. In this way, competition at each level is relatively fair and low selection pressure tends to allow young individuals to grow up and explore the new peaks sufficiently, while the implicit selection pressure enforced by the multiple fitness *levels* can still drive the evolutionary search in the desired direction. In HFC, individuals move from low-fitness subpopulations to higher-fitness subpopulations if and only if they exceed the fitness-based admission threshold of the receiving subpopulation, but not of a higher one.

In the HFC model (Figure 4.2), multiple subpopulations are organized in a hierarchy, in which each subpopulation can only accommodate individuals within a specified range of fitness. The entire range of possible fitnesses is spanned by the union of the subpopulations' ranges. Conceptually, each subpopulation has an admission buffer that has an *admission threshold* determined either initially (fixed) or adaptively. The admission buffer is used to collect qualified candidates, synchronously or asynchronously, from other subpopulations. Each subpopulation also has an *export threshold* (fitness level), defined by the admission threshold of the next higher-level subpopulation. Only individuals whose fitnesses are between the subpopulation's admission threshold and export threshold are allowed to stay in that subpopulation. Otherwise, they are exported to the appropriate higher-level subpopulation.

The HFC model was developed in order to improve evolutionary synthesis of bond graphs using genetic programming. In eigenvalue problems, we compared the performance of HFC-GP to standard GP and obtained the results in Hu *et al.*[23, 24]. The HFC technique dramatically improved on our previous results. HFC showed itself to be especially useful for the more difficult problems.

5. Case Study 1 - Eigenvalue Assignment

In this work, the main design objective is to find bond graph 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 genetic programming on this problem and to explore the topological and parametric behaviors of the bond graph models evolved.

The following three sets (consisting of 2, 4, and 6 target eigenvalues, respectively) were used as targets for example genetic programming runs:
{-1±2j}
{-1±2j, -2±j}

{-1±2j, -2±j, -3±0.5j}

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

$$Fitness\ (Eigenvalue\ ) = 0.5 + {}^{1}\!\Big/\!_{(2 + \sum Error\ / Order\ )}$$

A strongly-typed version (Luke[25]) of lilgp (Zongker and Punch[26]) is used to generate bond graph models. We obtained solutions with very small distance errors in a reasonable computation time. Detailed studies including several types of eigenvalue results and the effect of modifiable sites were reported in Seo *et al.*[27].

6. Case Study 2 – Analog Filter Design

6.1. Problem Definition

A filter design problem was used as a test of our approach for evolving electrical circuits with bond graphs, 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 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 target magnitudes; compute their differences and obtain the squared sum of differences as raw fitness. Then normalize the fitness according to:

$$Fitness\ (Filter\ ) = {}^{100}\!\Big/\!_{(100 + \sum Error\ )}$$

The GP parameters used for filter design were as follows:
Number of generations: 100
Population size: 300 in each of thirteen subpopulations and 2500 in each of two 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

6.2 Results

The performance of the evolved low-pass filter with a target cutoff frequency of 1000Hz is shown in Figure 6.2. Figure 6.3 gives the frequency response of an evolved high-pass filter with the same cut-off frequency. It shows that this result is also quite satisfactory. Figure 6.4 gives the frequency response of an evolved band-pass filter with cutoff frequencies at 10Hz and 1000Hz. Obviously, it is the most difficult of the three filter design problems. The statistical results of 10 runs each for high-, low- and band-pass filters are shown in Table 6.1. 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.

7. Case Study 3 – Printer Drive Redesign

7.1 Problem Definition

This example involves a drive system for a printer. The original problem was presented to one of the investigators by C. Denny and W. Oates of IBM, Lexington, KY, in 1972. Figure 2.3 (in section 2) shows a closed-loop control system to position a rotational load (inertia) denoted as J_L, and Figure 7.1 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.
The fitness function is defined as follows: within the time range of interest, uniformly sample 1000 points; compare the magnitudes of the step responses at the sample points with target magnitudes; compute their differences and obtain the squared sum of differences as raw fitness. Then normalize the fitness according to:

$$Fitness\ (\text{Printer}) = 1000 \big/ \left(1000 + \sum Error\right)$$

For comparison purposes, the performance simulation was first executed for the initial printer drive system. This design is found to be unsatisfactory because of unacceptable vibration of the load for 1700-1800ms.

Figure 7.1 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 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 bond graph model of the drive subsystem is given in Figure 7.2. 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 following cases were run on a single Pentium III 1Ghz PC with 256MB 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

7.2 Results

Several competing design candidates with different topologies resulted. One of the designs is shown in Figure 7.3. 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 embryo model (Figure 7.2). The performance of this model is shown in Figure 7.4. The position response for step function input quickly converges in 70msec, which was an acceptable timeframe. Detailed work was reported in Fan *et al.*[28].

8. Case Study 4 – Air Pump Design

8.1 Problem Definition

A schematic of an air pump is presented in Figure 8.1. It is a vibratory pump in which an electromagnetic circuit drives a small permanent magnet attached to a pivoted lever that, in turn, drives a rubber bellows pump. The bellows pump has rubber check valves and delivers a small flow of air. The basic structure of the air pump consists of the cascaded arrangement of three coupled subsystems: the electromagnetic actuator, the lever, and the air bellows. The average peak value of outflow for the original air pump is approximately 2×10^{-4} m^3/sec, as depicted in Figure 8.2.

The ultimate objective of the pump redesign is to maximize outflow subject to a constraint specifying maximum power consumption. The fitness function will consist of a positive term for pump outflow and a negative term for power consumption that is clamped at 0 so long as power is less than the constraint constant. However, the runs with the power constraint have not yet been performed, and the experimental results shown below are for an objective maximizing the outflow without implementing a power constraint.

The major GP parameters were as shown below:
Number of generations: 500
Subpopulation : 15
Population size: 200
Initial population: half_and_half
Initial depth: 3-6
Max depth: 15
Max nodes: 1200
Selection: tournament (size=7)
Crossover: 0.7
Mutation: 0.2
Reproduction: 0.1

8.2 Results

The GP program obtained satisfactory results on a Pentium-IV 1GHz in 5~15 minutes, which showed the efficiency of our approach in finding good design candidates. The algorithm kernel of HFC-GP was used.
Design variant 1 is represented in Figure 8.3. Three new components (two C, one TF), between the two dashed lines highlighted, were added at modifiable sites. Figure 8.4 displays outflow of this design variant. The average peak value of outflow is approximately 2.2×10^{-3} m^3/sec -- almost 9 times better than the original model, however, the power consumption is also somewhat higher than in the original design. The next step in the pump redesign process will be to specify a maximum power constraint and then seek designs that maximize the outflow produced subject to that power constraint. These runs have not yet been done.

9. Conclusion

This paper has suggested not only a new design methodology for automatically synthesizing designs for multi-domain, lumped parameter dynamic systems with a unified tool, but also describes new intermediate-complexity "atoms" for the synthesis process and two innovations in GP search methods. A careful combination of bond graphs and genetic programming, including careful attention to causality constraints in either of two ways, appears to be an appropriate approach to development of a method for efficient synthesis of complex multi-domain systems, such as mechatronic systems.
Four important improvements on our early work are introduced – imposing of regularity constraints on synthesized junction structures, higher-level causally consistent modular elements, structure fitness sharing, and hierarchical fair competition.

As proof of concept for this utility of these innovations, evolution of bond graphs for three different domain design problems – a specified-target-eigenvalues design, analog filter design, and printer drive redesign – was tested. Experiments showed that all three yielded satisfactory results in shorter times with small computational expense.

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. Also, experiments on the control of genetic operations on the modular primitives appear to offer promise, in the context of the enhanced search methods – SFS and HFC. Together, they appear to promise superior design solutions with modest computing resources.

Further study will aim at extension and refinement of the design methodology and application to design of more complex and inter-domain mechatronic systems.

Acknowledgments

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. Mechatronics*, vol. 1, no.1, 1996, pp. 29-38
[2] Karnopp, D. C., Rosenberg, R. C., Margolis, D. L. *System Dynamics, A Unified Approach*, 3nd ed., John Wiley & Sons, 2000.
[3] Rosenberg, R. C., Whitesell, J., Reid, J. "Extendable Simulation Software for Dynamic Systems," *Simulation*, *58*(3), 1992, pp.175-183.
[4] Rosenberg, R. C. "Reflections on Engineering Systems and Bond Graphs," *Trans. ASME J. Dynamic Systems, Measurements and Control*, v.115, 1993, pp.242-251
[5] Rosenberg, R. C. *The ENPORT User's Manual*, Rosencode Associate Inc., E. Lansing, MI, 1996.
[6] Rosenberg, R. C., Hales, M. K., Minor, M. "Engineering Icons for Multidisciplinary Systems," *Proc. ASME IMECE 1996*, DSC-V.58, 1996, pp.665-672.
[7] Sharpe, J. E., Bracewell, R. H. " The Use of Bond Graph Reasoning for the Design of Interdisciplinary Schemes," *1995 International Conference on Bond Graph Modeling and Simulation*, 1995, pp. 116-121.
[8] Youcef-Toumi, K., Glaviano, Y. A., Anderson, P. "Automated Zero Dynamics Derivation from Bond Graph Models," *1999 International Conference on Bond Graph Modeling and Simulation*, 1999, pp. 39-44
[9] Redfield, R. C. "Bond Graphs in Dynamic Systems Designs: Concepts for a Continuously Variable Transmission," *1999 International Conference on Bond Graph Modeling and Simulation*, 1999, pp. 225-230.
[10] Tay, E., Flowers, W., Barrus, J. "Automated Generation and Analysis of Dynamic System Designs," *Research in Engineering Design*, vol 10, 1998, pp. 15-29.
[11] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.
[12] Koza, J. R. *Genetic Programming II: Automatic Discovery of Reusable Programs*, The MIT Press, 1994
[13] Koza, J. R. Bennett, F. H., Andre, D., Keane, M. A. *Genetic Programming III, Darwinian Invention and Problem Solving*, Morgan Kaufmann Publishers, 1999.
[14] Koza, J. R, Bennett, F. H., Andre, D., Keane, M., Dunlap, A. "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Trans. Evolutionary Computation.*, *1*(2), 1997, pp.109-128.
[15] Holland, J. H. *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, E. D., Rosenberg, R. C., 2001, "First Steps toward Automated Design of Systems Using Bond Graphs and Genetic Programming", *Proc. Genetic and Evolutionary Computation Conference*, San Francisco, p. 189 (1-page abstract) and poster.
[18] Fan, Z., Hu, J., Seo, K., Goodman, E. D., Rosenberg, R. C., Zhang, B., 2001, "Bond Graph Representation and GP for Automated Analog Filter Design," *Genetic and Evolutionary Computation Conference Late-Breaking Papers*, San Francisco, pp. 81-86.
[19] DeJong, E. D., Watson, R. A., and Pollack, J. B., "Reducing Bloat and Promoting Diversity using Multi-Objective Methods," *Proc. GECCO-2001*, Morgan Kaufmann, San Francisco, pp. 11-18.
[20] Ekárt, A., S. Németh, Z., "A Metric for Genetic Programs and Fitness Sharing," in R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, T. Fogarty (eds.), *Genetic Programming, Proceedings of EUROGP'2000*, Edinburgh, 15-16 April 2000, LNCS volume 1802, pp. 259-270.
[21] McKay, R. I., "Fitness Sharing in Genetic Programming," *Proc. GECCO-2000*, Las Vegas, NV, Morgan Kaufmann, San Francisco, July, 2000.
[22] Hu, J., Seo, K., Li, S., Fan, Z., Rosenberg, R. C., Goodman, E. D., "Structure Fitness Sharing (SFS) for Evolutionary Design by Genetic Programming," Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002, New York, July, 2002, pp. 780-787.

[23] Hu, J., Goodman, E. D., "Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms," CEC 2002, Honolulu, Hawaii, May, 2002.
[24] Hu, J., Goodman, E. D., Seo, K., Pei, M., "Adaptive Hierarchical Fair Competition(AHFC) Model for Parallel Evolutionary Algorithms," *Proc. Genetic and Evolutionary Computation Conference,* GECCO-2002, New york, July, 2002, pp. 772-779.
[25] Luke, S. 1997, *Strongly-Typed, Multithreaded C Genetic Programming Kernel,* http://www.cs.umd.edu/users/-seanl/gp/patched-gp/.
[26] Zongker, D., Punch, W.F. 1998, *lil-gp 1.1 User's Manual*, GARAGe, College of Engineering, Michigan State University.
[27] Seo, K., Hu, J., Fan, Z., Goodman, E. D., Rosenberg, R. C. , "Automated Design Approaches for Multi-Domain Dynamic Systems Using Bond Graphs and Genetic Programming," The International Journal of Computers, Systems and Signals, vol.3, no.1, pp.55-70, 2002.
[28] Fan, Z., Seo, K., Rosenberg, R. C., Hu, J., Goodman, E. D., "Exploring Multiple Design Topologies using Genetic Programming and Bond Graphs", Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002, New York, July, 2002, pp. 1073-1080.

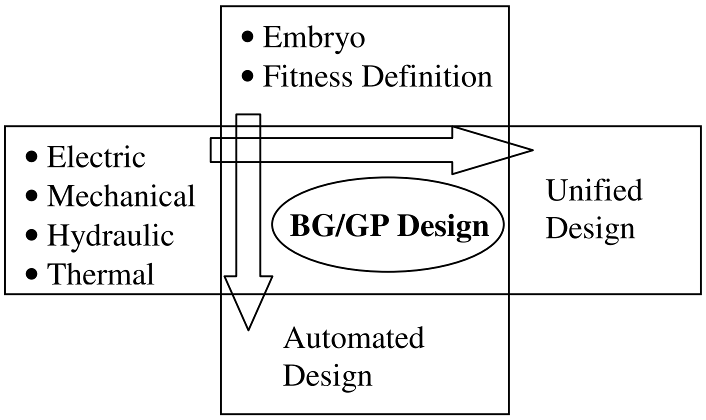


Figure 2.1. Key features of the BG/GP design methodology

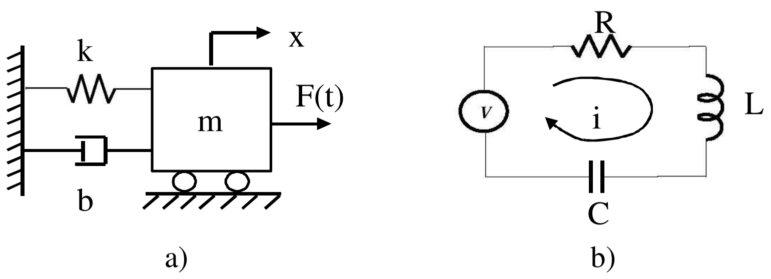


Figure 2.2. Example single-domain systems: a) mechanical, and b) electrical

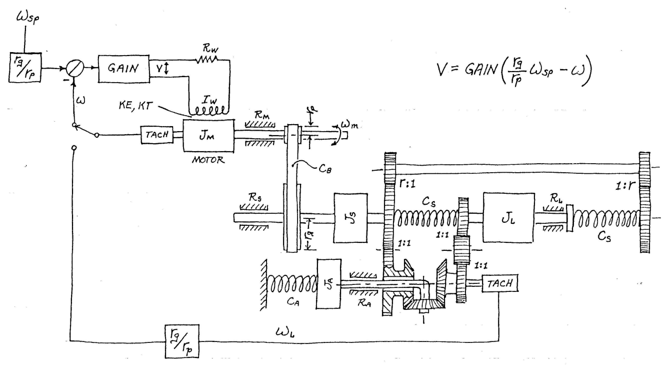


Figure 2.3. Schematic diagram of an example mechatronic system – the printer drive

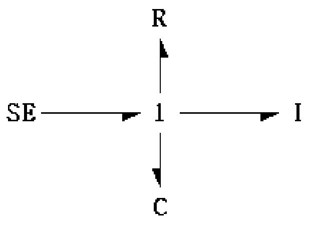


Figure 2.4. Bond graph model for Figure 2a) and 2b)

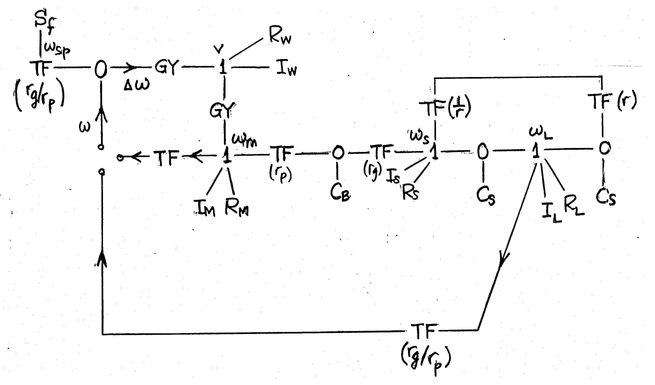
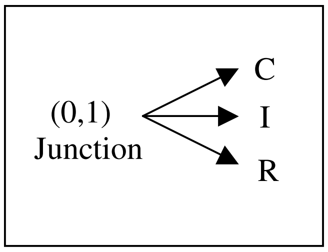
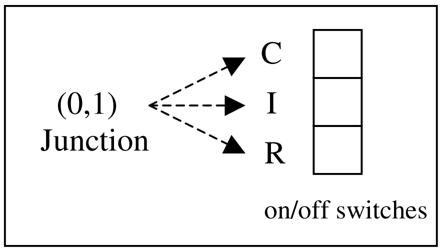


Figure 2.5. Bond graph model for printer drive of Figure 2.3



(a)



(b)

Figure 3.1. a) Causally well-posed set, b) Complex set

Table 3.2 Comparison between *basic* set and *causally well-posed* set

6-Eigenvalue Placement Problem						
	Basic set			Causally well-posed set with regular junctions		
	Fitness	Average Distance error	Computation Time (min)	Fitness	Average Distance error	Computation Time (min)
1	0.965716	0.147	115	0.993611	0.026	40
2	0.964347	0.154	97	0.996270	0.015	25
3	0.963244	0.159	81	0.996991	0.012	76
4	0.963970	0.155	72	0.979393	0.086	50
5	0.944992	0.247	98	0.998337	0.007	44
6	0.947936	0.232	87	0.998510	0.006	70
7	0.989103	0.045	55	0.997527	0.010	49
8	0.966533	0.143	76	0.999464	0.002	43
9	0.968623	0.134	90	0.995881	0.017	61
10	0.978255	0.091	110	0.981709	0.076	39
Avg	0.965272	0.151	88.1	0.993769	0.026	49.7
S.D.	0.012771	0.059	18.06	0.007175	0.030	15.38

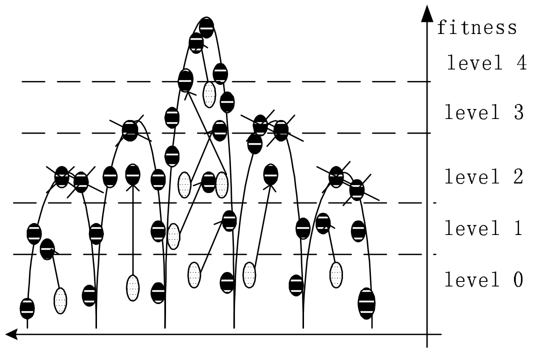


Figure 4.1. Hierarchical Competition

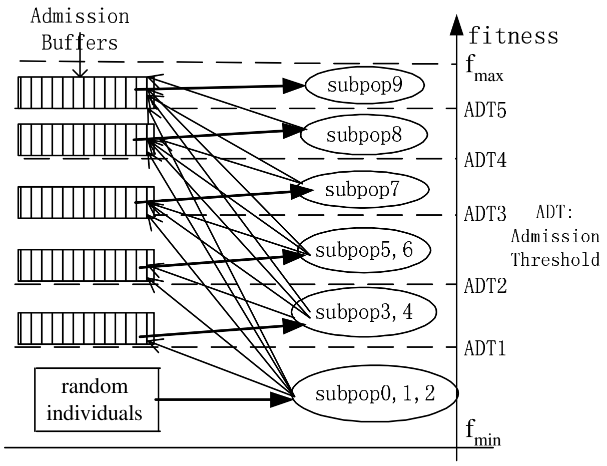


Figure 4.2 Hierarchical fitness level, multiple subpopulations

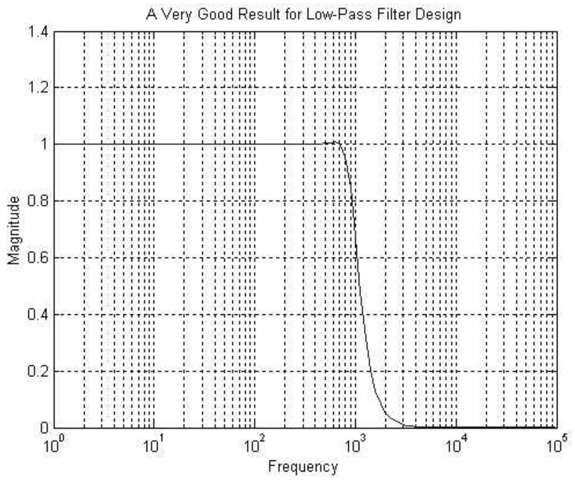


Figure 6.2 Frequency response of evolved low pass filter

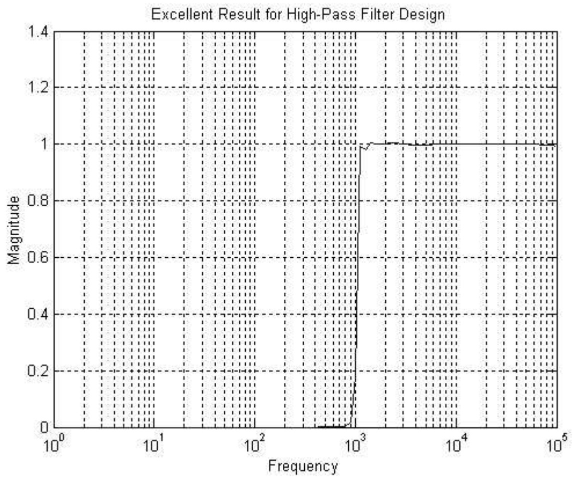


Figure 6.3 Frequency response of evolved high-pass filter

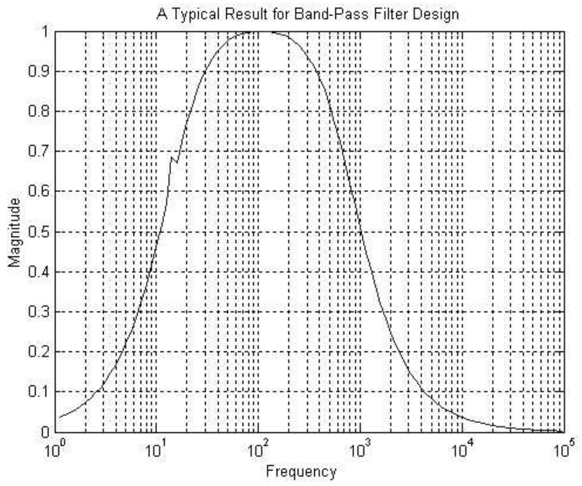


Figure 6.4 Frequency response of evolved band pass filter

Table 6.1 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
Avg	2.670	0.974017	1.963	0.980827	11.407	0.898045
S.D	0.530	0.00502	0.936	0.008994	2.541	0.021033

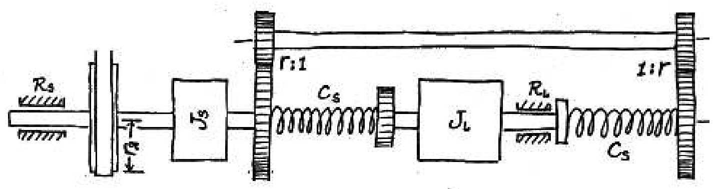


Figure 7.1 The initial printer drive subsystem.

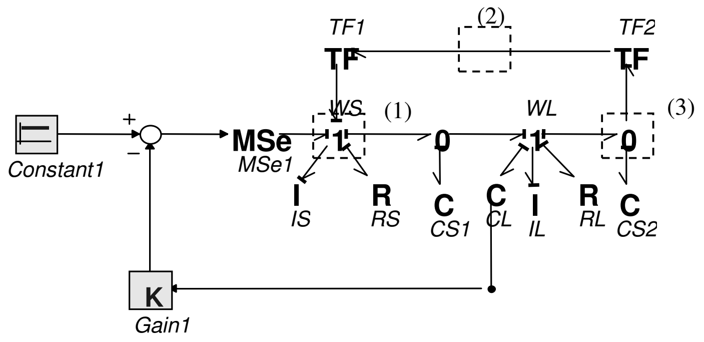
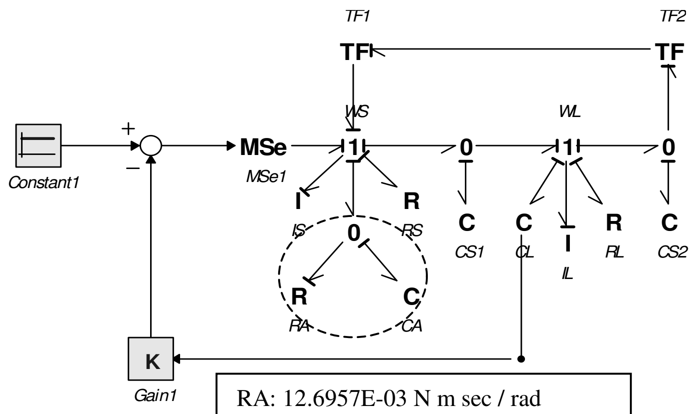


Figure 7.2 The embryonic bond graph model with feedback loop



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

Figure 7.3. An evolved bond graph model

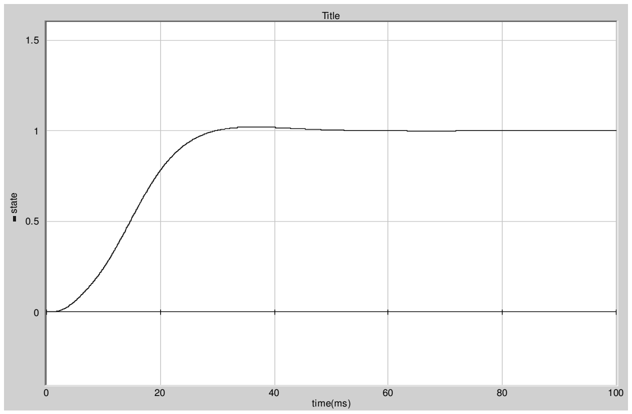


Figure 7.4 Simulation result of evolved bond graph model

Table 7.1 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
Avg	15.358	0.984875
S.D	0.6903	0.000669

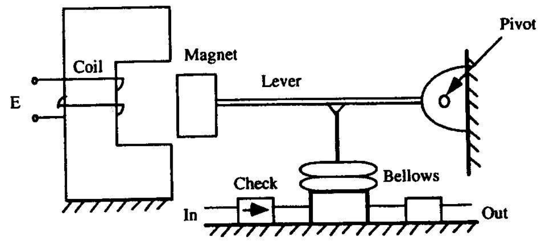


Figure 8.1 Schematic of air pump model

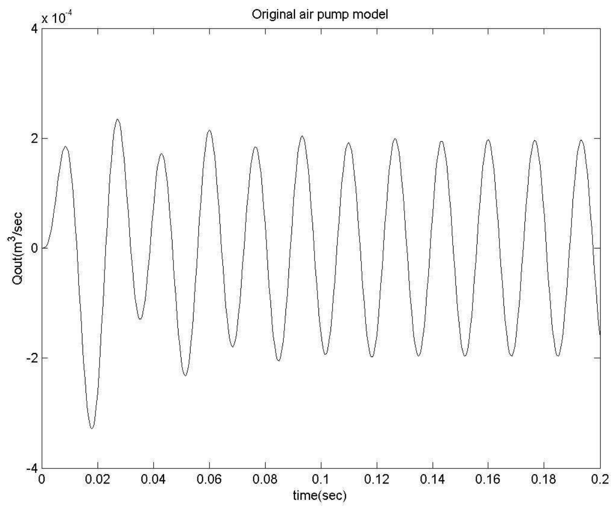


Figure 8.2 Outflow of original air pump model

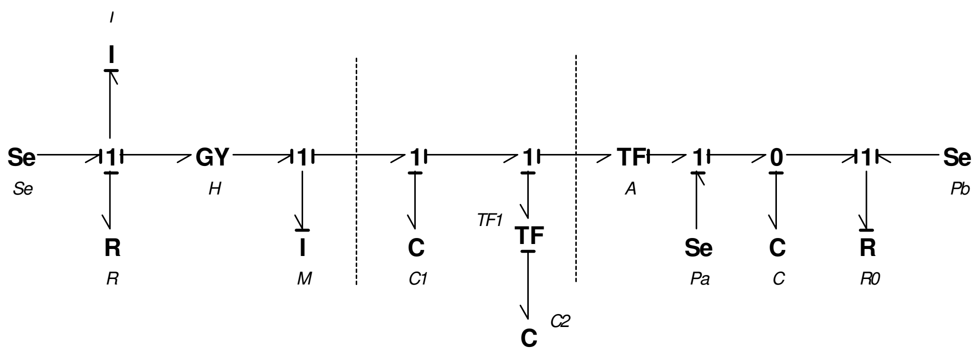


Figure 8.3 Bond Graph model of design variant 1 for air pump

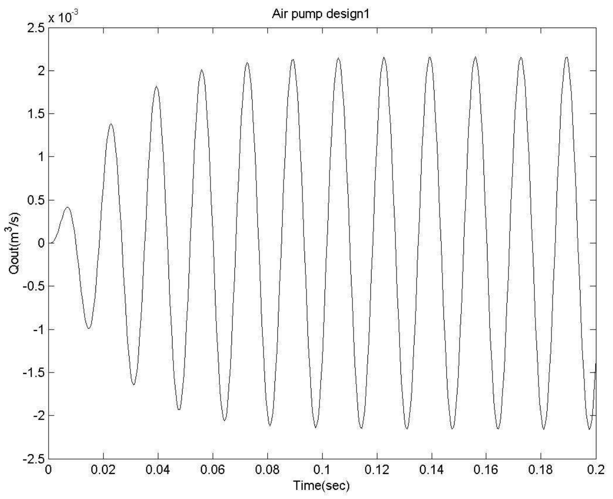


Figure 8.4 Outflow of design variant 1 for air pump