Exploring Open-Ended Design Space of Mechatronic Systems

CHAPTER · JULY 2005

Source: InTech

3 AUTHORS, INCLUDING:

Zhun Fan
Shantou University
84 PUBLICATIONS **581** CITATIONS
SEE PROFILE

Erik D. Goodman
Michigan State University
188 PUBLICATIONS **2,573** CITATIONS
SEE PROFILE

Exploring Open-Ended Design Space of Mechatronic Systems

Zhun Fan, Jiachuan Wang & Erik Goodman

1. Introduction

In general, design of mechatronic 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 mechatronic systems. The strategy is to develop an automated procedure capable of exploring the search space of candidate mechatronic 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 evolutionary computation approach.
For example, automated design of analog circuits has attracted much attention in recent years (Grimbleby, 1995) (Lohn, 1999)(Koza, 1999)(Fan, 2000). They 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 amount of components to be evolved (Lohn, 1999). Although their work basically achieve good results in analog circuit design, they are 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., 1998).

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 artefacts. It is closely related to our approach, except that it searches in a single energy domain.

To develop an integrated mechatronic design environment, we investigate an approach combining genetic programming and bond graphs to automate the process of design of mechatronic 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 typewriter. Several design alternatives for the typewriter drive are derived through exploring open-topologies in bond graph space.

2. Design Domain and Methodology

2.1 Mechatronic Systems and Bond Graph Representations

The reason we used bond graphs in research on mechatronic system synthesis is because mechatronic systems are intrinsically multi-domain systems. We need a uniform representation of mechatronic systems so that designers can not only shift among different hierarchies of design abstractions but also can move around design partitions in different physical domains without difficulty.

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 components, etc. It is the explicit representation of model topology that makes the bond graph a good candidate for use in open-ended design search.

Fig. 1. shows an example of a single bond graph model that represents a resonator unit in both mechanical domain and electrical domain.

In addition to appropriate "drivers" (sources), the left part of Fig. 1. shows a lumped-parameter dynamical mechanical system model typically including a mass, spring and damper while the right part of Fig.1. shows an "RLC" electric circuit including a resistor, inductor and capacitor. However, they could both be expressed in the same bond graph shown in the middle of Fig.1 because both the mechanical and electrical subsystem share the same dynamic behavior and governed by the same dynamic equations.

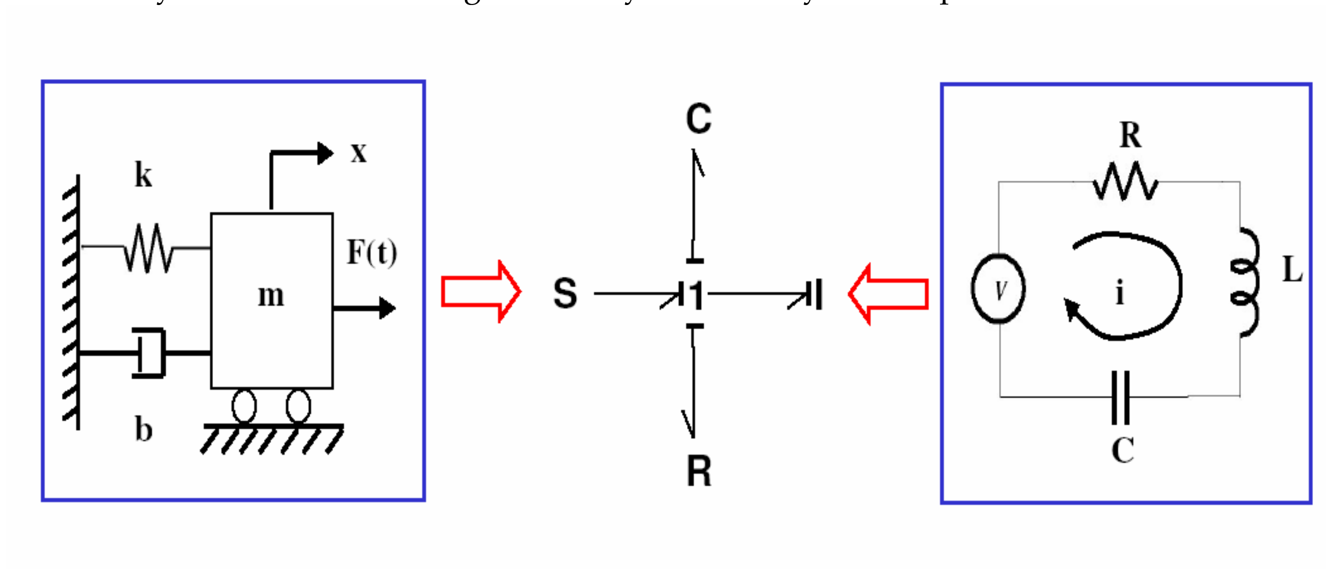


Figure 1. Bond graph representation of dynamic systems

It is also very natural to use bond graphs to represent a dynamic system, such as a mechatronic system, with cross-disciplinary physical domains and even controller subsystems (Fig. 2.)

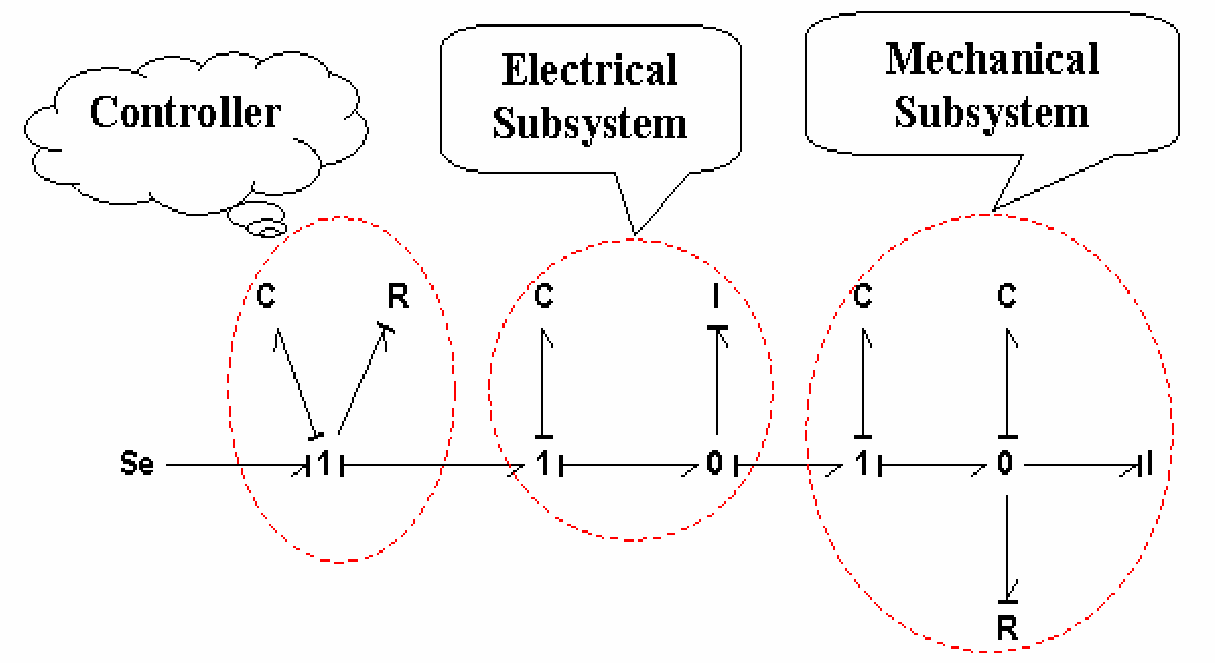


Figure 2. Bond graph representing a mechatronic system with mixed energy domains and a controller subsystem

2.2 Bond Graph

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 & Bracewell, 1995, Tay et al., 1998, Youcef-Toumi, 1999, Redfield R., 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 are simulatable, 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 graphs, 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 mechatronic system design space in an evolutionary process.

2.3 Combining Genetic Programming and Bond Graph

The most common form of genetic programming (Koza, 1994) uses trees to represent the entities to be evolved. 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).

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)

Table 1. Function and terminal set for bond graph evolution

Defining a proper function set is one of the most significant steps in using genetic programming. It may affect both the search efficiency and validity of evolved results and is closely related to the selection of building blocks for the system being designed. In this work, the genotypes assembled from the function sets are constructors which, upon execution, specify a bond graph.

In other words, when the genotype is executed, it generates the phenotype in a developmental manner. In this research, we have an additional dimension of flexibility in generating phenotypes, because bond graphs are used as modeling representations for multi-domain systems, serving as an intermediate representation between the mapping of genotype and phenotype, and those bond graphs can be interpreted as systems in different physical domains, chosen as appropriate to the circumstances.

Fig. 3 gives a particular example in the domain of electrical circuits and Fig. 4 illustrates the role of bond graphs in the mappings from genotypes to phenotypes (Fan et al., 2001)

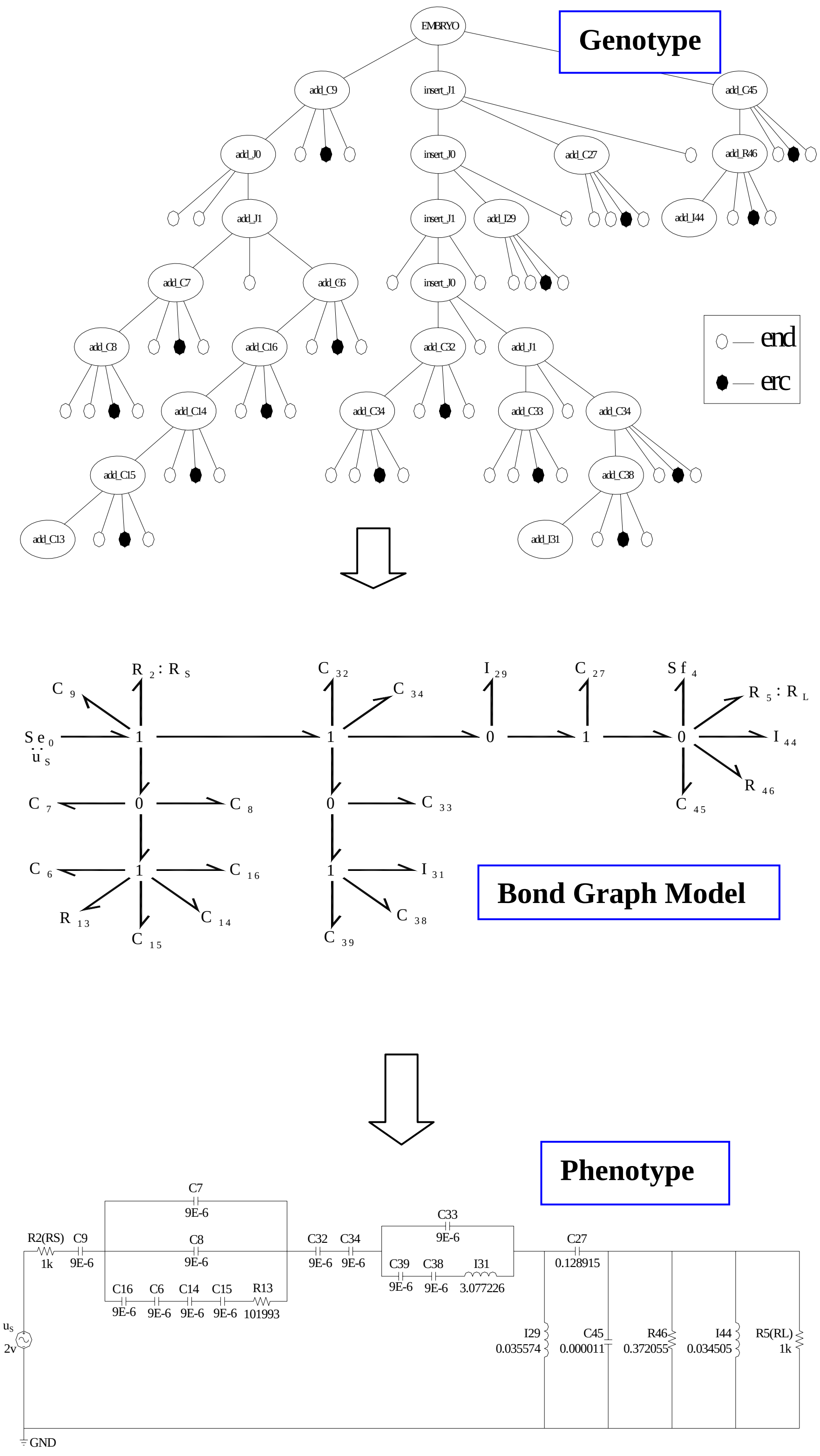


Figure 3. Example of Genotype-Phenotype Mapping in the Electrical Circuit Domain

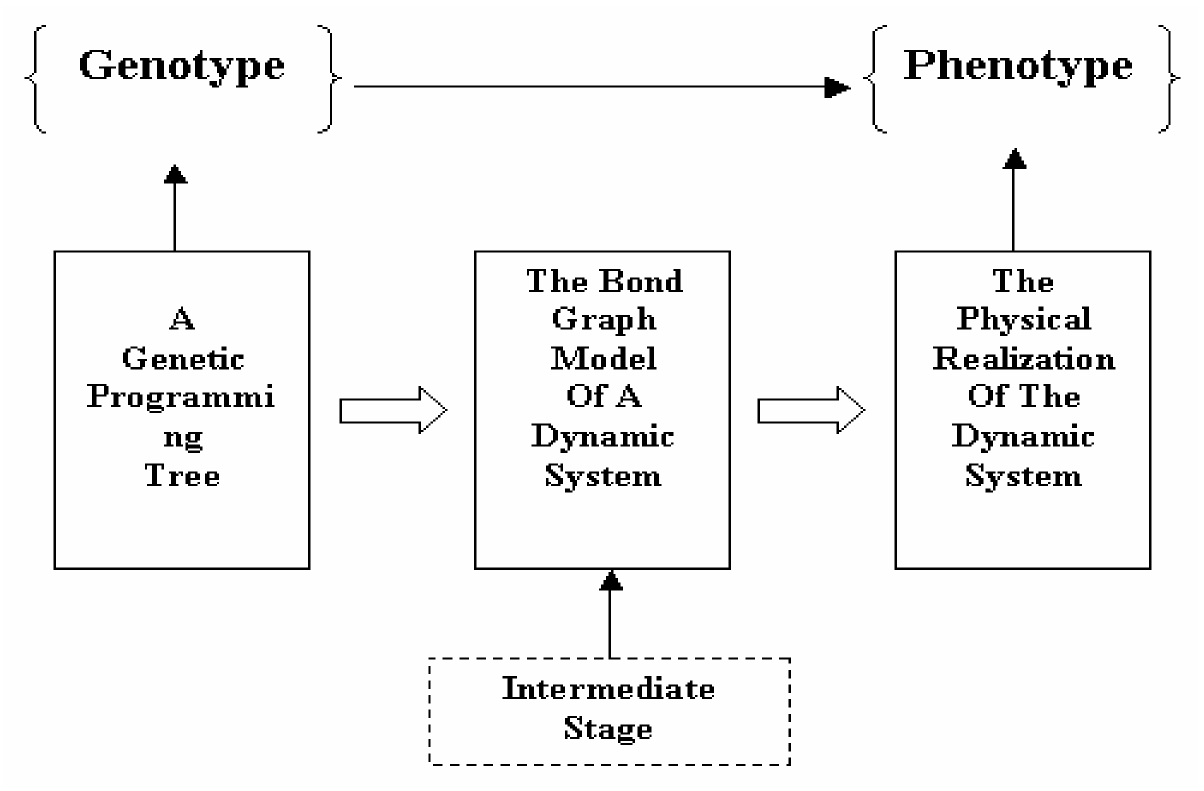

Genotype
Phenotype
A Genetic Programmi ng Tree
The Bond Graph Model Of A Dynamic System
The Physical Realization Of The Dynamic System
Intermediate Stage

Figure 4. Genotype-Phenotype mapping

2.4 Design Procedure

The flow of the entire algorithm is shown in Fig. 5. Based on a preliminary analysis, 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 each 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 specified number of iterations is performed. The final step is to instantiate a physical design, replacing the bond graphs with physical components it represents.

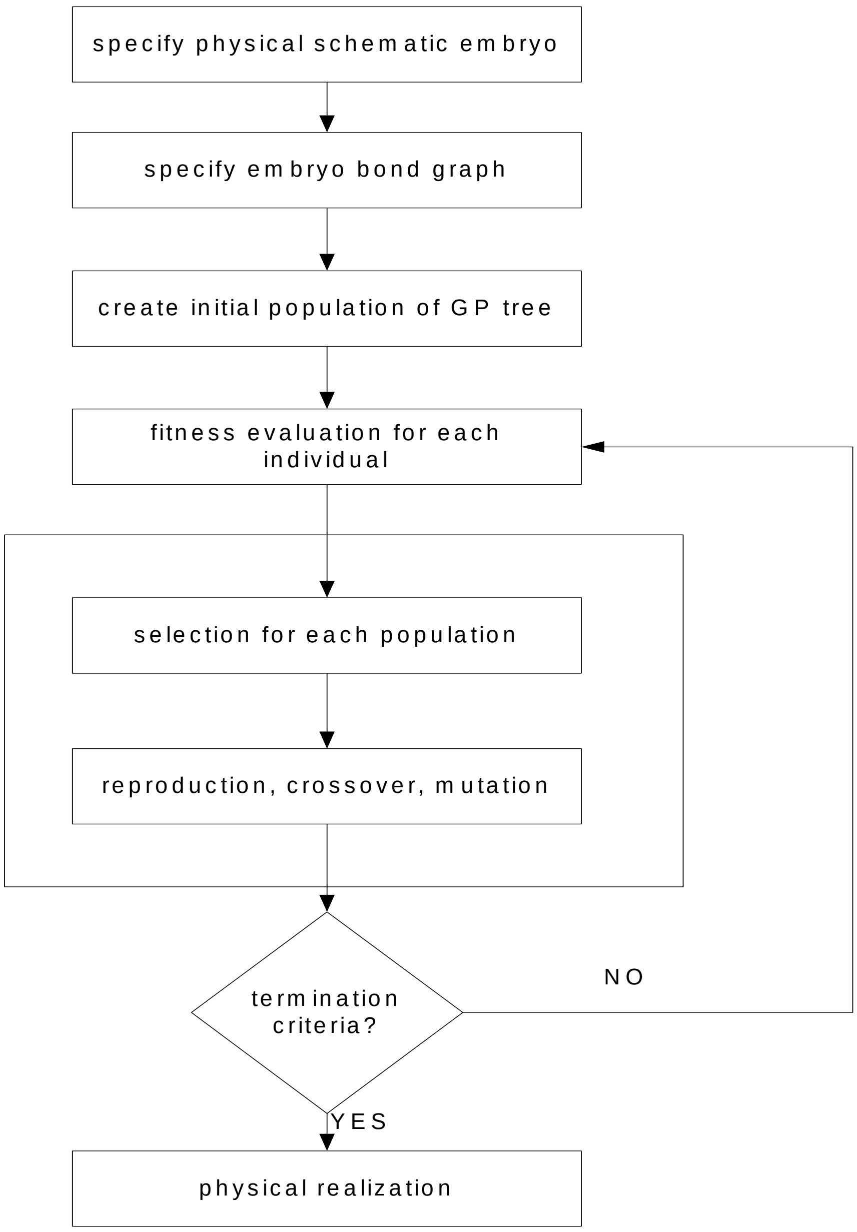

specify physical schematic embryo
specify embryo bond graph
create initial population of GP tree
fitness evaluation for each individual
selection for each population
reproduction, crossover, mutation
termination criteria?
NO
YES
physical realization

Figure 5. Flow chart of the design procedure

2.5 Integrated Evolutionary Mechatronics Synthesis

Because the final target of this research is to improve the quality of strategic and early design decisions, enhance human-computer cooperation, and ultimately reduce product and overall system life cycle cost, an integrated design and synthesis framework for mechatronic systems is presented and to be investigated, to cover a full spectrum of customer needs and product life considerations.
Fig. 6. provides a graphical overview of the integrated design environment (Wang, 2004). Evolutionary computation techniques, including genetic programming, are utilized to explore the open-ended design space for mechatronic systems as the core high-performance computing algorithm. Bond Graphs are used to unify representations of mechatronic subsystems from different domains. Design performances can be evaluated both through time domain simulation and via frequency domain analysis.

The design primitives at both conceptual and physical realization levels are stored in the design repository so that designers can retrieve them either manually or through classifying schemes. The Graphical User Interface (GUI) is designed to better understand customer needs, through the specification of design representation, requirements and constraints interactively. It can also incorporate user preferences under different trade-off strategies. The process of synthesis and analysis are iterative until design process converges to satisfactory design solutions.

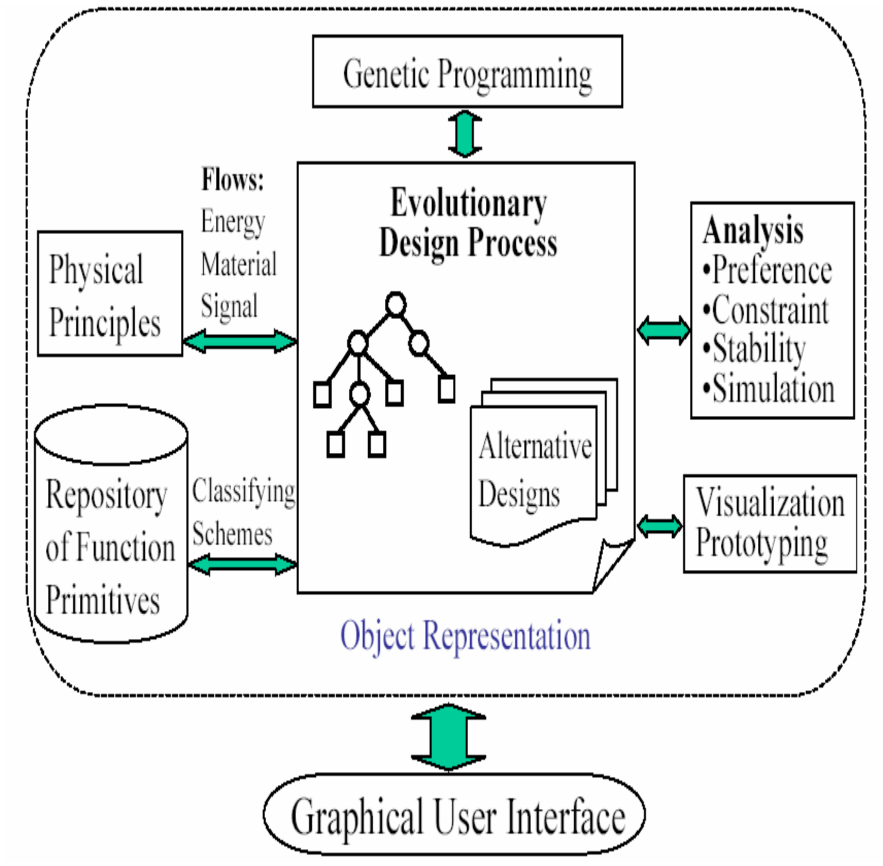


Figure 6. Integrated mechatronics design environment

3. Case Study

3.1 Problem Formulation

The original problem was presented by C. Denny and W. Oates of IBM, Lexington, KY, in 1972. Fig. 7. shows a closed-loop control system to position a rotational load (inertia) denoted as JL.

The system includes electric voltage source, motor and mechanical parts. As it is a multi-domain mechatronic system, a bond graph is convenient to use for modelling (see Fig. 8a). The problem with the design is the position output of the load JL has intense vibrations (see Fig. 8b).

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 Fig. 8b is 4000 ms.

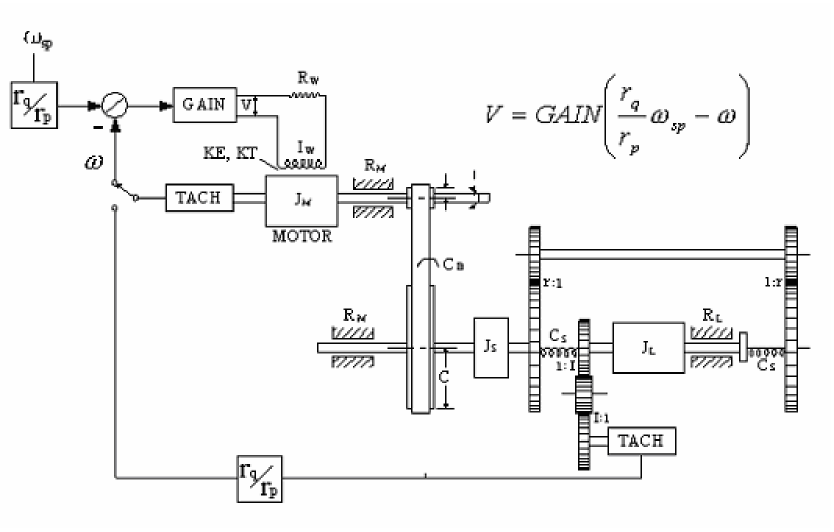
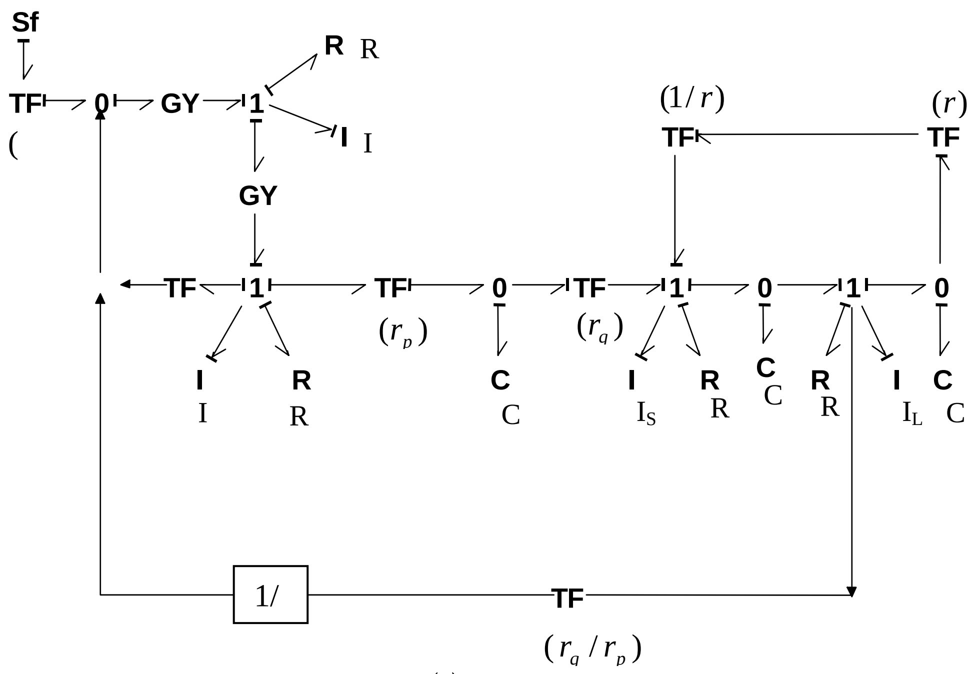
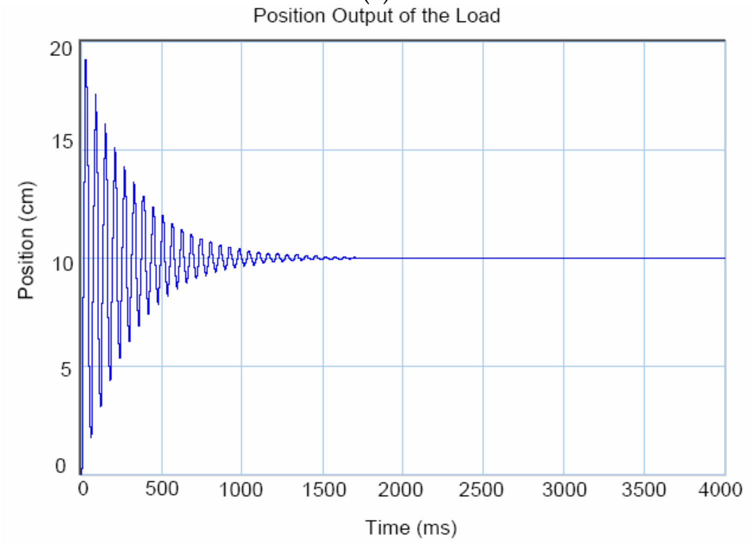


Figure 7. Schematic of the typewriter drive system



(a)



(b)

Figure 8. a) Bond graph model b) Positional vibration of the load

3.2 Embryo of Design

By analysing the model, we conclude that the critical part for the design is a subsystem that involves the drive shaft and the load (see Fig. 9). The input is the driving torque, Td, generated through the belt coupling back to the motor (not shown).

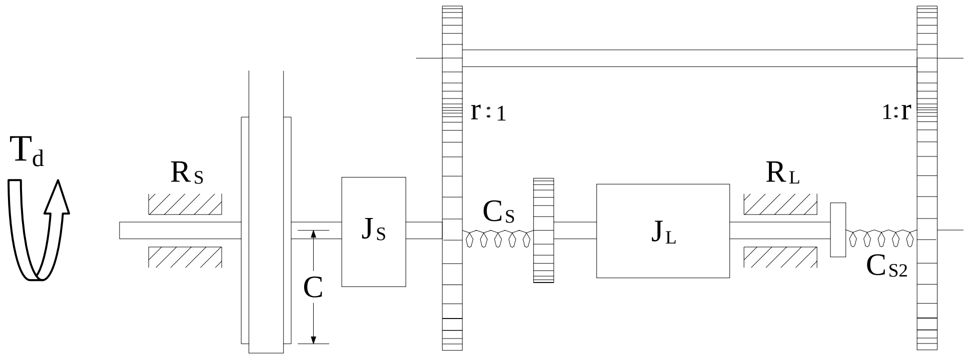


Figure 9. The embryo subsystem

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.
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 Fig. 10, 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. Note that modifiable sites are only possible spots for insertion of new components; the search may not use all 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.

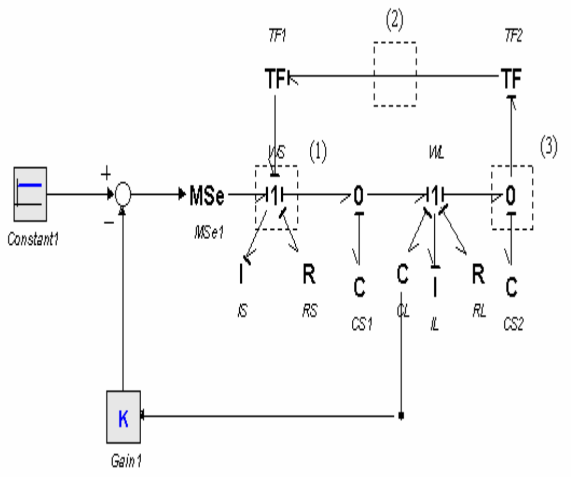


Figure 10. The embryo subsystem

The parameters for the embryo model are:

$I_s : 6.7 \times 10^{-6}\ kg \cdot m^2$

$R_s : 0.013 \times 10^{-3}\ N \cdot m \cdot \sec \diagup rad$

$C_{s1} : 0.208\ N \cdot m \cdot \diagup rad$

$C_{s2} : 0.208\ N \cdot m \cdot \diagup rad$

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

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

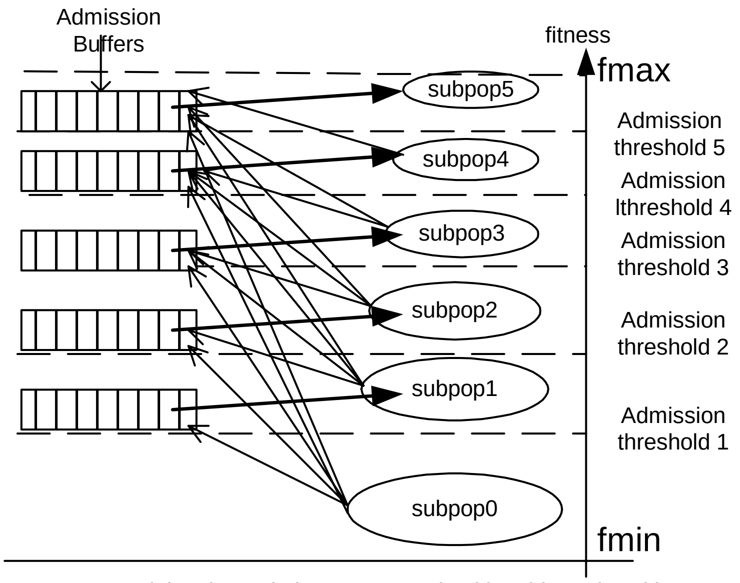
$C_L : 1.0 \times 10^{6}\ N \cdot m \cdot \diagup rad$

$TF1 : 0.1, \quad TF2 : 10$

For simplicity and without loss of generality, both gains of K and MSe are set to be unit.

3.3 The Hierarchical Fair Competition (HFC) Model

A special form of genetic programming is applied in this research. In HFC (Hierarchical Fair Competing) model (Fig. 11), 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 HFC model balances exploration and exploitation of GP effectively. Our experience shows that using the HFC model can also substantially increase the topology diversity of the whole population and help to provide the designer a diverse set of competing design candidates for further trade-offs.



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 11. Hierarchical Fair Compete model of GP

3.4 Definition of Fitness Function

The fitness function of individual design is defined according to the position output response of the load JL 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 JL at the sample points with target magnitudes (unity in this example), compute their difference and get a squared sum of difference as raw fitness, defined as $Fitness_{raw}$. Then normalized fitness is calculated according to:

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

It can be assumed approximately that the higher the normal 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 achieved design results (Fig. 12-18) 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 & Punch, 1996) to generate bond graph models.

The major GP parameters were as shown below:

Number of generations: 100
Population sizes: 200 in each of 15 subpopulations
Initial population: half_and_half
Initial depth: 3-6 Max depth: 17
Selection: Tournament (size=7)
Crossover: 0.9
Mutation: 0.1

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 software package 20Sim was used to verify the dynamic characteristics of the evolved design.

3.6 Experimental Observations

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.

Ten runs of this problem have been done and most of the runs produced satisfactory solutions.
The fitness history of a typical run is shown in Fig. 12. Two competing design candidates with different topologies, as well as their performances, are provided in Fig. 13 to Fig. 18 (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 70ms. Note that the time scale of the plots is 100 ms.
One of the designs is shown in Fig. 13. 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 (Fig. 13). Dashed circles highlight the newly evolved components in the bond graph figures. The performance of this model is shown in Fig. 14.
The position response for step function input quickly converges in about 50msec, which was an acceptable timeframe. One possible physical realization of the bond graphs model is shown in Fig. 13. A spring and a damper are added and coupled to the original printer subsystem as shown in Fig. 14. Another design is shown in Fig. 16.
Four elements, 0-junction with C, 1-junction with R are added to modifiable site 2 and one R is added to modifiable site 3. One possible physical realization of the design is shown in Fig. 17. Fig. 18 displays the performance of this model.Table 2 represents the statistical results of 10 runs for the printer drive.
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.

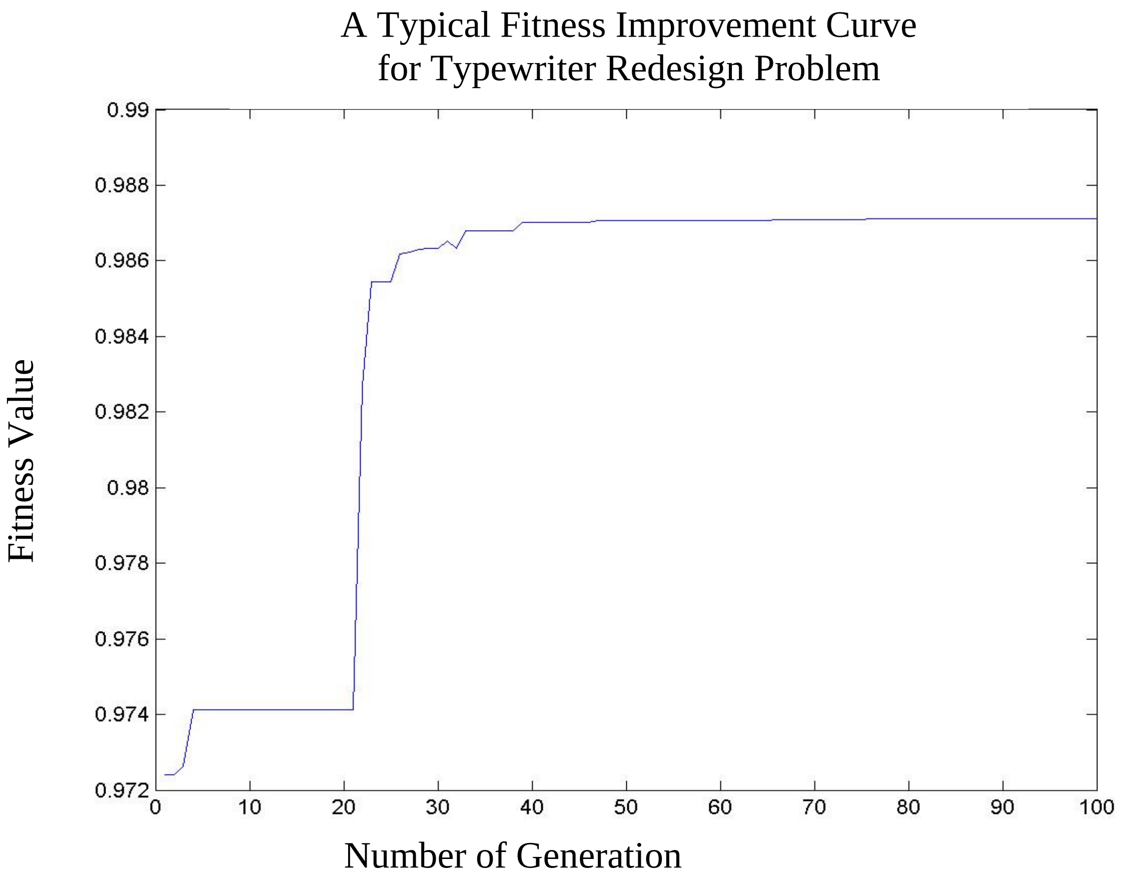


Figure 12. Fitness history for a typical typewriter drive redesign run

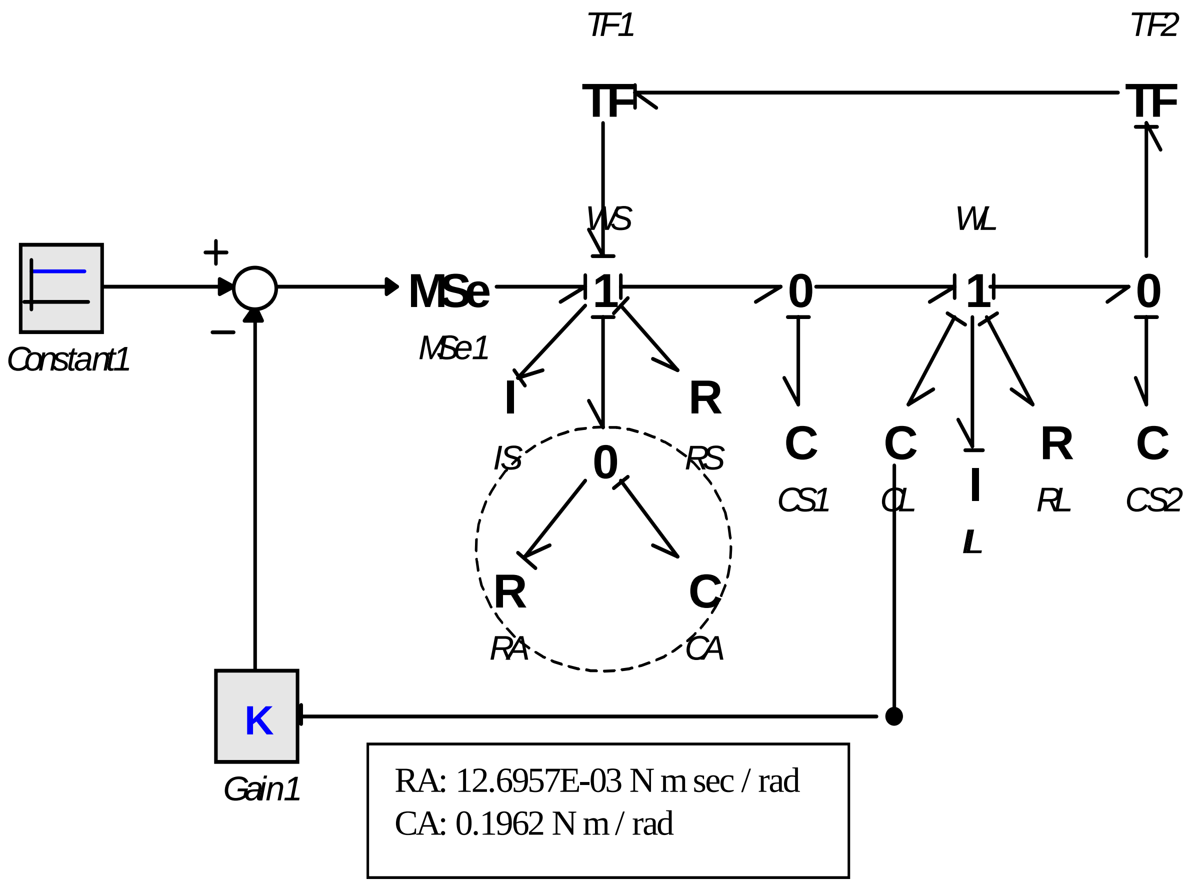


Figure 13. The evolved bond graph model I

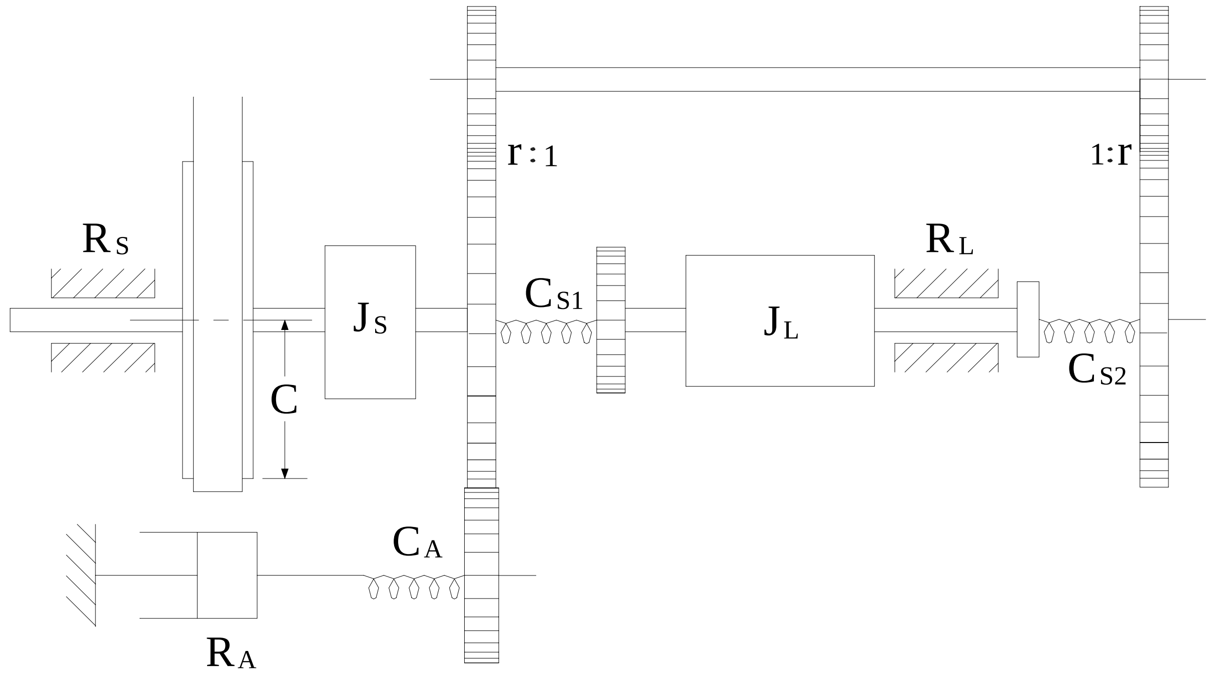


Figure 14. The physical realization of evolved bond graph model I

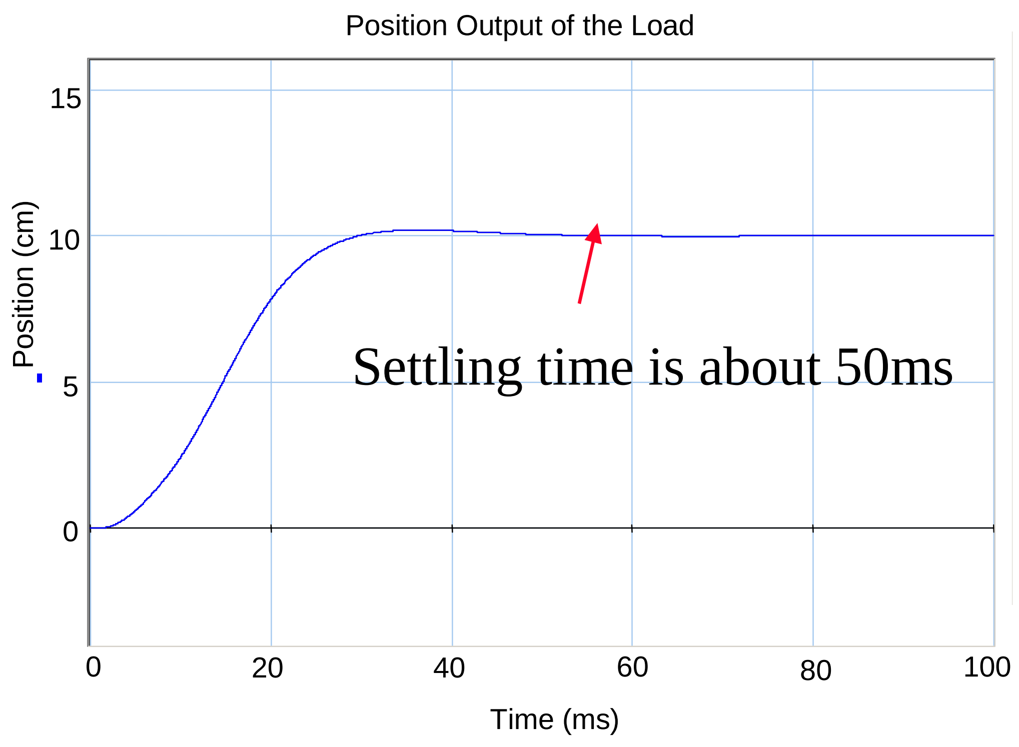


Figure 15. Simulation result of evolved bond graph model I

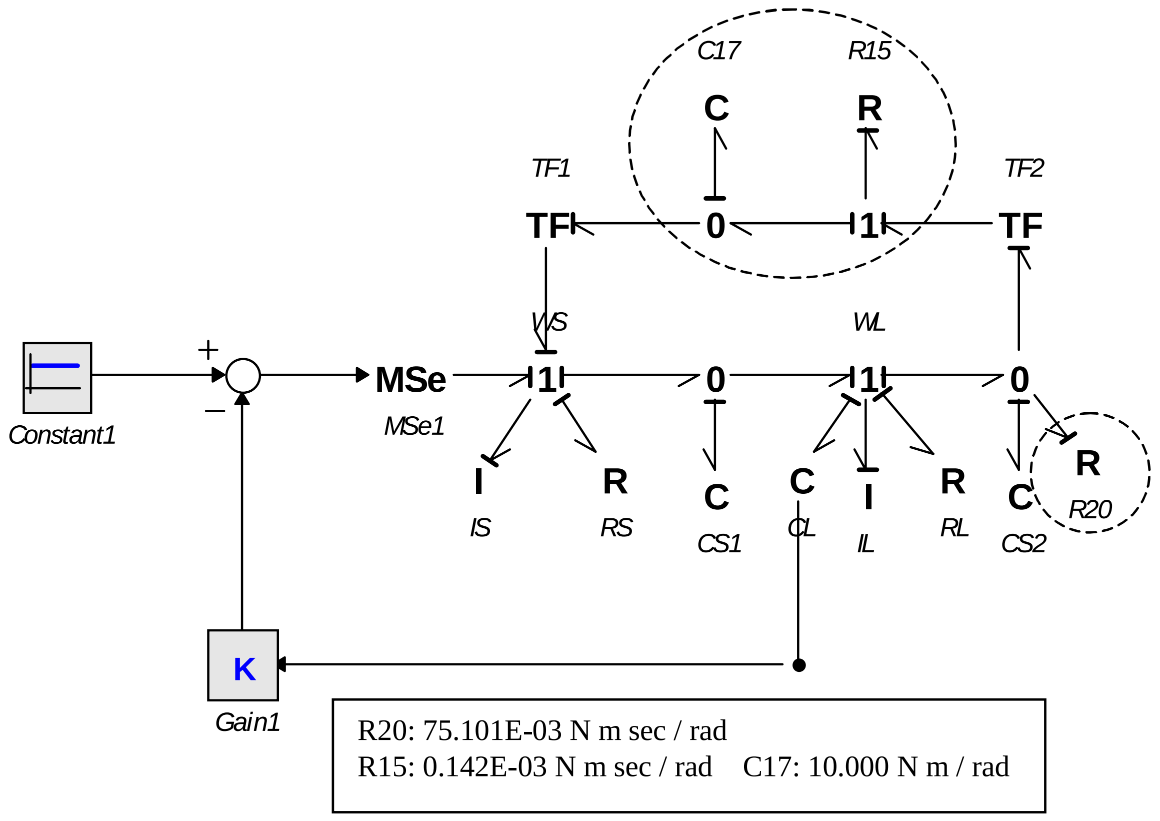

C17
R15
C
R
TF1
TF2
TF
0
1
TF
WS
WL
+
−
Constant1
MSe
MSe1
1
0
1
0
I
IS
R
RS
C
CS1
C
CL
I
IL
R
RL
C
CS2
R
R20
K
Gain1
R20: 75.101E-03 N m sec / rad
R15: 0.142E-03 N m sec / rad   C17: 10.000 N m / rad

Figure 16. The evolved bond graph model II

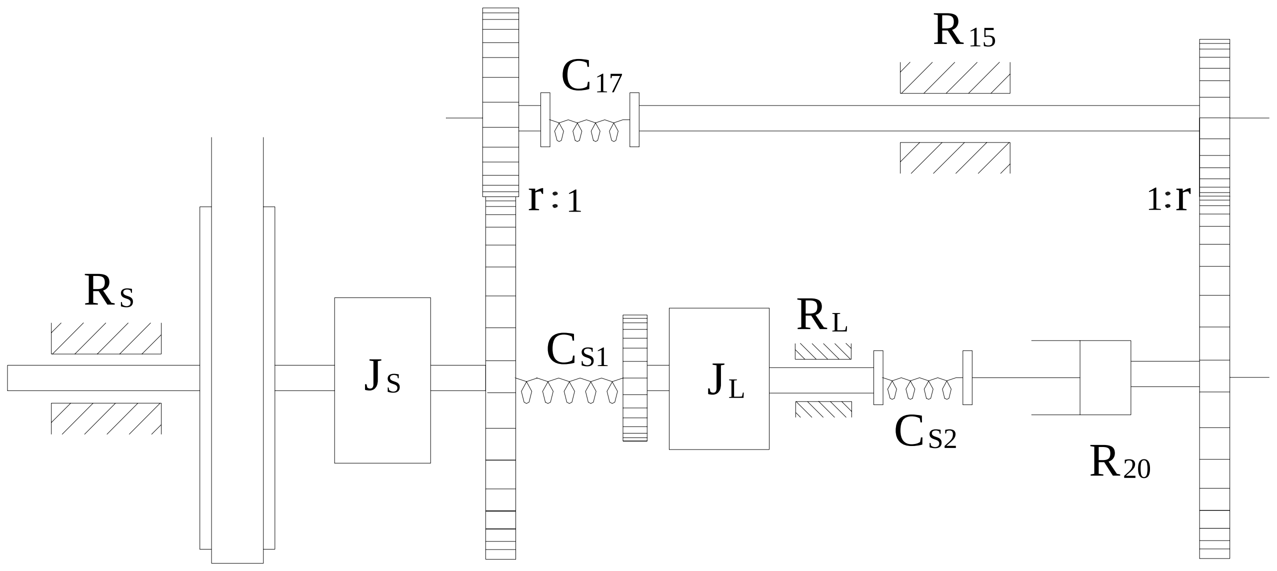


Figure 17. The physical realization of evolved bond graph model II

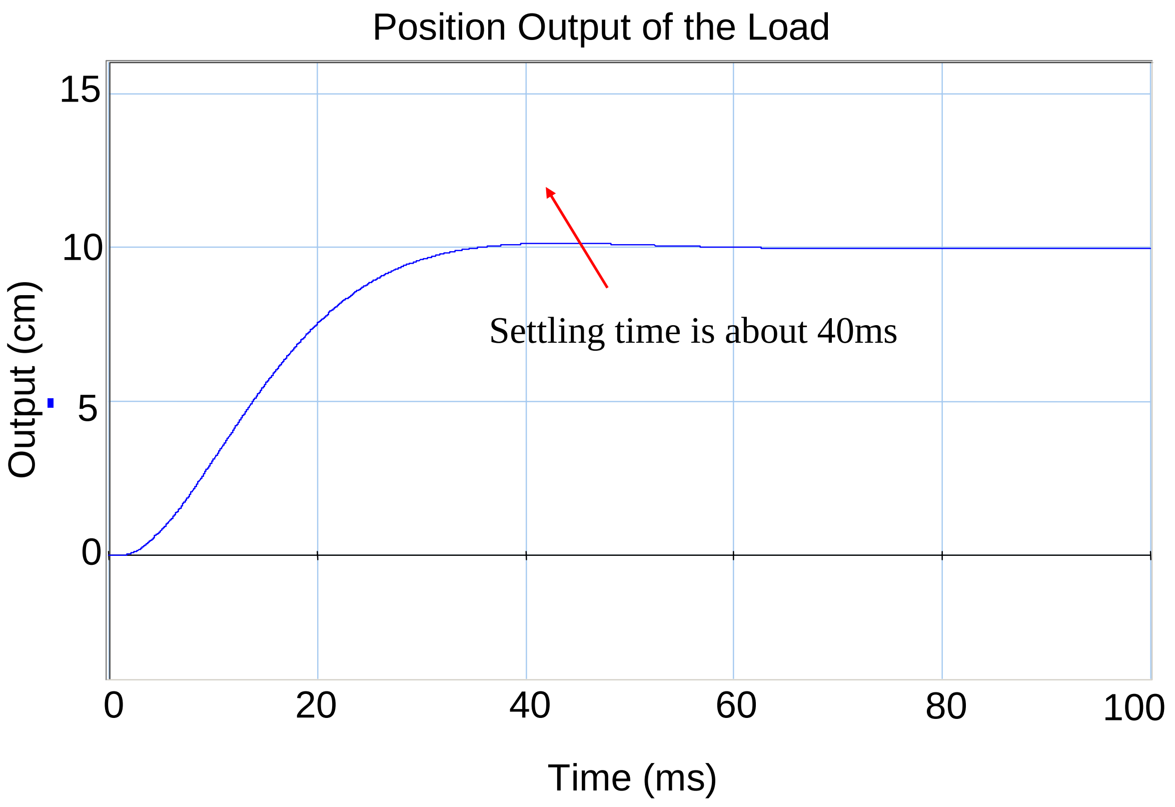


Figure 18. Simulation result of evolved bond graph model II

Run No.	Fitness of Printer	
	Distance	Fitness
1	15.076	0.985
2	15.818	0.984
3	15.188	0.985
4	16.720	0.983
5	15.053	0.985
6	14.085	0.986
7	15.122	0.985
8	15.502	0.985
9	15.132	0.985
10	15.881	0.984
Best	14.085	0.986
Worst	16.720	0.984
Average	15.358	0.985
Standard Deviation	0.6903	0.000669

Table 2. Summary results of fitness for printer

4. Conclusions

This research has explored a new automated approach for synthesizing designs for mechatronic systems. By taking advantage of genetic programming as a search method for competent designs and the bond graph as a representation for mechatronic systems, we have created a design environment in which open-ended topological search can be accomplished in a semi-automated and efficient manner and the design process thereby facilitated. By incorporating specific design considerations the method can be used to explore design space of special types of mechatronic systems such as robotic systems.

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

Our long-term target in this research is to design an integrated and interactive synthesis framework for mechatronic systems that covers the full spectrum of design processes, including customer needs analysis, product development, design requirements and constraints, automated synthesis, design verification, and life-cycle considerations.

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

5. References

Coelingh H. ; T.J.A. de Vries & van Amerongen J. (1998). Automated Performance Assessment of Mechatronic Motion Systems during the Conceptual Design Stage. Proc. 3rd Int'l Conf. on Adv. Mechatronics, pp. 472-477, Okayama, Japan.

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

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

Hu J., Goodman E. D., Seo K., Pei M., (2002). Adaptive Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002, New York, pp. 772-779.

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

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

Koza J., Bennett F. III, Andre D. & Keane M. (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.

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

Luke S., 1997, Strongly-Typed, Multithreaded C Genetic Programming Kernel, available from http://cs.gmu.edu/~sean/research/lil-gp-patch/ . Accessed: 2005-01-15

Paynter H. (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.

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

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

Rosenberg R., Whitesell J., & Reid J. (1992). Extendable Simulation Software for Mechatronic Systems. Simulation Vol. 58, pp. 175-183.

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

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

Tay E., Flowers W. & Barrus J. (1998). Automated Genration and Analysis of Mechatronic System Designs. Research in Engineering Design, Vol. 10, pp. 15-29.

Wang J. (2004). Integrated Coevolutionary Synthesis of Mechatronic Systems Using Bond Graphs. PhD dissertation, Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst

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

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

Zongker D. and Punch W., III, (1998), lil-gp 1.1 User's Manual, GARAGe, College of Engineering, Michigan State University, available from http://garage.cps.msu.edu/. Accessed: 2005-01-15