Dense and Switched Modular Primitives for Bond Graph Model Design

Kisung Seo[1], Zhun Fan[1], Jianjun Hu[1], Erik D. Goodman[1], and Ronald C. Rosenberg[2]

[1]Genetic Algorithms Research and Applications Group (GARAGe)
Michigan State University
[2]Department of Mechanical Engineering, Michigan State University
East Lansing, MI 48824, USA
{ksseo,fanzhun,hujianju,goodman,rosenber}@egr.msu.edu

Abstract. This paper suggests dense and switched modular primitives for a bond-graph-based GP design framework that automatically synthesizes designs for multi-domain, lumped parameter dynamic systems. A set of primitives is sought that will avoid redundant junctions and elements, based on pre-assembling useful functional blocks of bond graph elements and (optionally) using a switched choice mechanism for inclusion of some elements. Motivation for using these primitives is to improve performance through greater search efficiency and thereby to reduce computational effort. As a proof of concept for this approach, an eigenvalue assignment problem, which is to find bond graph models exhibiting minimal distance errors from target sets of eigenvalues, was tested and showed improved performance for various sets of eigenvalues.

1 Introduction

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 (Youcef-Toumi [1]). However, most current research for evolutionary design has been optimized for a single domain (see, for example, Koza et. al., [2,3]).

In order to overcome this limitation and enable open-ended search, the Bond Graph / Genetic Programming (BG/GP) design methodology has been developed, based on the combination of these two powerful tools (Seo *et al.* [4,5] and tested for a few applications – an analog filter (Fan *et al.* [6]), printer drive mechanism (Fan et. al., [7]), and air pump design (Goodman *et al.* [8]). BG/GP worked efficiently for these applications. The search capability of this system has been improved dramatically by introduction of a new form of parallel evolutionary computation, called Hierarchical Fair Competition GP (HFC-GP, Hu, *et al*., [9]), which can strongly reduce premature convergence and enable scalability with smaller populations.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1764–1775, 2003.

However, two issues still arise: one is the need for much stronger synthesis capability arising from the complex nature of multi-domain engineering design, and the other is the desire to minimize computational demands. While we have made inroads in improving of GP search by introducing HFC-GP, we want to exploit the notion of modularity of GP function primitives to make additional gains. Much useful modularity can be discovered during an evolutionary process, as is done, for example, by the ADF (Koza [10]). However, in many cases, we believe that explicit introduction of higher-level modules as function primitives, based on domain knowledge, will yield faster progress than requiring their recognition during the evolutionary process. Some research has been devoted to choice or refinement of the function set in GP. Soule and Heckendorn [11] examined how the function set influences performance in GP and showed some relationship between performance and GP functions sets, but their work was limited to generating simple sine functions varying only arithmetic and trigonometric operators (e.g., +, -, *, /, tan, ….). We will try to exploit higher-level function sets, rather than simply choosing different sets at the same level.

In this paper, a generic type of primitive is introduced, and specialized here to capture specific domain knowledge about bond graphs – the *dense switched modular primitive.*

First, we introduce the *dense* module concept to generate compact bond graph models with fewer operations. It replaces several operations in the basic (original) set with one operation, yielding a smaller tree attainable with less computational effort.

Second, the *switched* module concept creates a small function set of elements with changeable forms, which can assist in evolving complex functionality, while eliminating many redundant bond graph structures evolved if it is not used. Elements eliminated include “dangling” junctions that connect to nothing and many one-port components (such as resistors, capacitors, inductors, etc.). Their elimination makes the resulting bond graph simpler and the speed of evolution faster.

A careful design of a dense and switched modular primitive should considerably increase the efficiency of search and also, for the bond graph case, the efficiency of fitness assessment, as is illustrated in this paper.

As a test class of design problems, we have chosen one in which the objective is to realize a design having a specified set of eigenvalues. The eigenvalue assignment problem is well defined and has been studied effectively using linear components with constant parameters. Section 2 discusses the inter-domain nature, efficient evaluation, and graphical generation of bond graphs, including the design methodology used in approaching such problems. Section 3 explains the basic set and redundancy problem and Sect. 4 describes the dense switched modular primitive set. Section 5 presents results for 6-, 10- and 16-eigenvalue design problems, and Sect. 6 concludes the paper.

2 Evolutionary Bond Graph Synthesis for Engineering Design

2.1 The BG/GP Design Methodology

There is a strong need for a unified design tool, able to be applied across energy domains – electrical, mechanical, hydraulic, etc. 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. Automation of this process – so the user sets up the specifications and "pushes a button," then receives candidate design(s) – is also important.

A design methodology that combines bond graphs and genetic programming can serve as an automated and unified approach (Fig. 1). The proposed BG/GP (Bond Graph with Genetic Programming) design methodology requires only an embryo model and fitness (performance) definition in its initial stage; the remaining procedures are automatically executed by genetic programming search. However, due to the complexity of the engineering design problem, the need for efficiency in the design search is very high. It is this problem that is addressed here.

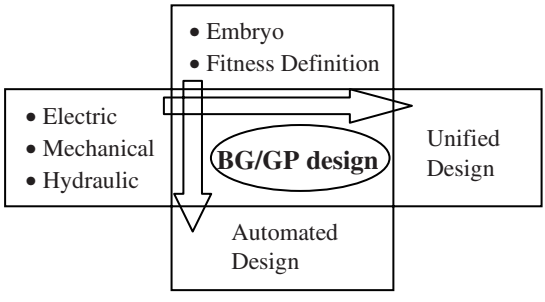

Fig. 1. Key features of the BG/GP design methodology

2.2 Bond Graphs

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. The middle of Fig. 2 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(left), or an RLC electrical circuit. S_e corresponds with force in mechanical systems, or voltage in electrical (right). 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.

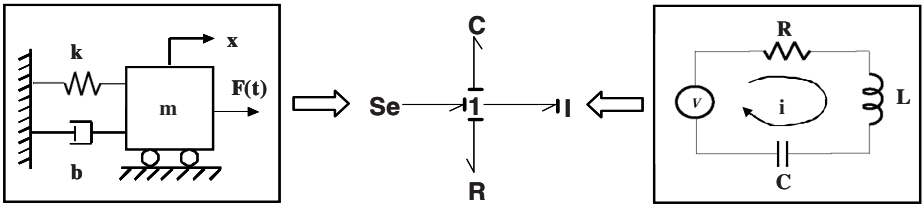

Fig. 2. The same bond graph model for two different domains

3 Basic Set and Redundancy

The initial BG/GP system used GP functions and terminals for bond graph construction as follows. There are four types of functions: *add* functions that can be applied only to a junction and which add a C, I, or R element; *insert* functions that can be applied to a bond and which insert a 0-junction or 1-junction into the bond; *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 *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*. [5].

Table 1. Functions and terminals in Basic set

Name	Description
add_C	Add a C element to a junction
add_I	Add an I element to a junction
add_R	Add an R element to a junction
insert_J0	Insert a 0-junction in a bond
insert_J1	Insert a 1-junction in a 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
+	Sum 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)

Many redundant or unnecessary junctions and elements were observed in experiments with this basic set. Such unnecessary elements can be generated by the free combinatorial connection of elements, and, while they can be removed without any change in the physical meaning of the bond graph, their processing reduces the efficiency of

processing and of search. At the same time, such a "universal" set guarantees that all possible topologies can be generated. However, many junctions "dangle" without further extension and many arrangements of one-port components (C, I, R) that can be condensed are generated. Figure 3 illustrates redundancies that are marked with dotted circles in the example. First, the dangling 0- and 1-junctions in the left-hand figure can be eliminated, and then three C, I, and R elements can be joined together at one 1-junction. Furthermore, two R elements attached to neighboring 0-junctions can be merged to a single equivalent R. Avoiding these redundant junctions and elements improves search efficiency significantly.

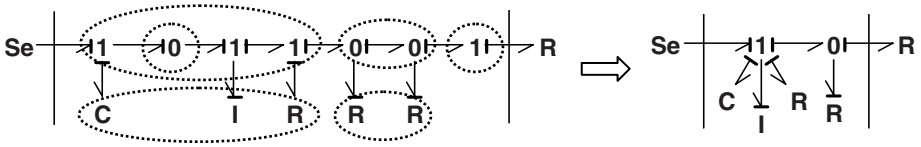

Fig. 3. Example of redundant 0- and 1-junctions and R elements (left) in generated bond graph model, and equivalent model after simplification (right). The dotted lines represent the boundary of the embryo

4 Construction of Dense Switched Modular Primitives

The redundancy problem is closely related with the performance and computational effort in the evolutionary process. The search process will be hastened by eliminating the redundancy, and it is hypothesized that this will happen without loss of performance of the systems evolved. It is obvious that computational resources can be saved by removal of the redundancy. To reduce the redundancy noted above and to utilize the concept of modularity, a new type of GP function primitives has been devised – the *dense switched* modular primitives ("DSMP"). Roughly speaking, a *dense* representation (eliminating redundant components at junctions, guaranteeing causally well-posed bond graphs, and avoiding adjacent junctions of the same type) will be combined with a *switched* structure (allowing components that do not impact causal assignment at a junction to be present or absent depending on a binary switch).

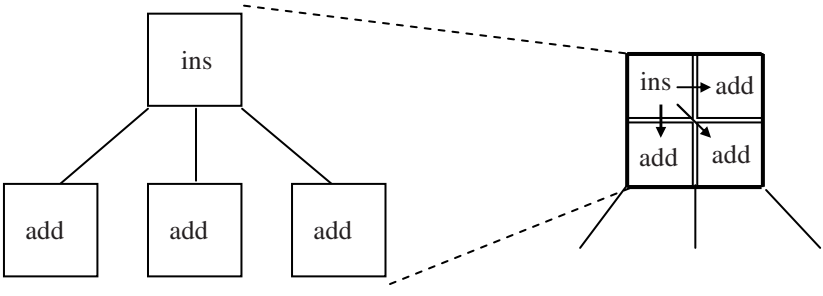

Fig. 4. The dense modular primitive

The major features of the modular primitives are as follows. First, a single dense function replaces all add, insert, and replace functions of the basic set. This concept is explained in Fig. 4, in which mixed *ins* and *add* operations can be merged into one operation. Therefore, a GP tree that represents a certain bond graph topology can be much smaller than attainable with the basic set. This dense function not only incorporates multiple operations, but also reflects design knowledge of the bond graph domain, such as causality (discussed later).

Second, any combination of C, I, and R components can be instantiated according to the values of a set of on/off *switch* settings that are evolved by mutation. This modularity also helps to relieve the redundancy of C, I, and R components, giving them fewer places to proliferate that appear to be different, but are functionally equivalent. This new set introduces further modularity through a controllable switching function for selection of C, I, R combinations (Fig. 5). The function set of the dense switched modular primitives is shown in Table 2. It consists of two functions that replace all *ins*, *add*, and *replace* functions in the basic set (Table 1).

Table 2. New functions in the switched modular primitive set

Name	Description
insert_JPair_SWElements	Insert a 0-1 (or 1-0) junction pair in a bond and attach switched C, I, R elements to each junction
add_J_ SWElements	Add a counter-junction to a junction and attach switched C, I, R elements

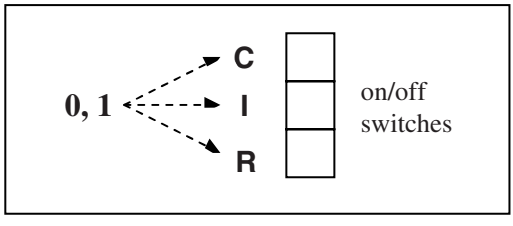

Fig. 5. Switched modular primitive

Third, the proper typing of 0-junctions and 1-junctions is determined by an implicit genotype-phenotype mapping, considering the neighbor junction to which the primitive is attached. This allows insertion of only "proper pairs" of junctions on bonds, preventing generation of consecutive junctions of the same type that are replaceable by a single one.

Fourth, we insure that we generate only feasible individuals, satisfying the causally well-posed property, so automatic state equation formulation is simplified considerably. One of the key advantages of BG/GP design is the efficiency of the evaluation. The evaluation stage is composed of two steps: 1) causality analysis, and, when merited, 2) dynamic simulation. The first, causal analysis, allows rapid determination of feasibility of candidate designs, thereby sharply reducing the time needed for analysis of designs that are infeasible. In most cases, all bonds in the graph will have been assigned a causal stroke (determining which variables are assigned values at that point, rather than bringing to it pre-assigned values) using only integral causality of C or I and extension of causal implication. Some models can have all causality assigned without violation – the causally satisfied case. Other models are assigned causality, but with violations – the causally violated case. If one has to continue to use an arbitrary causality of an R, it means that some algebraic relationships must be solved if the equations are to be put into standard form. This case can be classified as causally undetermined. Detail causality analysis is described in Karnopp *et al*. [12].

The dense switched modular primitives with implicit genotype-phenotype mapping and the guaranteed feasibility of the resulting causally well-posed bond graphs can speed up the evolution process significantly.

5 Experiments and Analysis

To evaluate and compare the proposed approach with the previous one, the eigenvalue assignment problem, for which the design objective is to find bond graph models with minimal distance errors from a target set of eigenvalues, is used. 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.

5.1 Problem Definition

In the example that follows, a set of target eigenvalues is given and a bond graph model with those eigenvalues must be generated, in a classic "inverse" problem. The following sets (consisting of various 6-, 10- and 16-eigenvalue target sets, respectively) were used for the genetic programming runs:

- Eigenvalue sets used in experiments:
  1) {-1±2j, -2±j, -3±0.5j}
  2) {-10±j, -1±10j, -3±3j }
  3) {-20±j, -1±20j, -7±7j}
  4) {-1, -2, -3, -4, -5, -6}
  5) {-20±j, -1±20j, -7±7j, -12±4j, -4±12j }
  6) {-1, -2, -3, -4, -5, -6, -7, -8, -9, -10}
  7) {-20±1j, -1±20j, -7±7j, -12±4j, -4±12j, -15±2j, -9±5j, -5±9j}

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. Relative distance error (normed by the distance of the target from the origin) is used.

$$Fitness(Eigenvalue) = 0.5 + 1/(2 + \sum Error/Order)$$

We used a strongly-typed version (Luke, [13]) of lilgp (Zongker and Punch [14]) with HFC (Hierarchical Fair Competition, Hu, *et al*., [9]) GP to generate bond graph models. These examples were run on a single Pentium IV 2.8GHz PC with 512MB RAM. The GP parameters were as shown below.

Number of generations : 500
Population sizes : 100 in each of ten subpopulations for multiple population runs
Initial population: half_and_half
Initial depth : 3-6
Max depth : 12 (with 800 max_nodes)
Selection : Tournament (size=7)
Crossover : 0.9
Mutation : 0.1

The tabular results of 6- and 10-eigenvalue runs are provided in Tables 3-4, with statistics including mean relative distance error (averaged across each target eigenvalue) and mean tree size, for each set of 10 experiments.

Table 3 illustrates the comparison between the basic set and the DSMP (dense switched modular primitive) set on typical complex conjugate and real six-eigenvalue target sets. In the first set, {-1±2j, -2±j, -3±0.5j}, the average error of the basic set (0.151) is larger than that of the DSMP set (0.043). The second and third sets, for two different target eigenvalue sets that have larger norms from the origin, show average distance errors of the basic set that are also larger. The numbers in parentheses regarding distance error of the DSMP set represent their ratio to the basic set distance errors.

Table 3. Results for 6 eigenvalues

6-Eigenvalue Placement Problem (10 runs)				
	Basic set		DSMP set	
Eigenvalue set	Dist error	Tree Size	Dist error	Tree Size
{-1±2j, -2±j, -3±0.5j}	0.151	513.6	0.043(28%)	237.0
{-10±1j, -1±10j, -3±3j}	0.068	451.8	0.026(38%)	296.8
{-20±1j, -1±20j, -7±7j}	0.056	399.4	0.021(37%)	285.6
{-1, -2, -3, -4, -5, -6}	0.144	445.7	0.009(6%)	307.1

In a fourth example, an all-real set of target eigenvalues {-1, -2, -3, -4, -5, -6} is tested and shows that the ratio of errors between the approaches is more than ten (0.144 for the basic set vs. 0.009 for the DSMP set, only 6% of the basic set error). Also, mean tree sizes of all basic set runs are much larger than those of DSMP set.

Results for a 10-eigenvalue assignment problem are shown in Table 4. The results for a complex conjugate 10-eigenvalue set {-20±1j, -1±20j, -7±7j, -12±4j, -4±12j} show that the average error of the basic set (0.210) is three times larger than that of the DSMP set (0.064). The results for a real 10-eigenvalue set also show the average error of the basic set (0.267) is more than ten times larger than that of the DSMP set (0.023). As with 6 eigenvalues, the mean tree sizes of the basic set are larger than those of the DSMP set.

Table 4. Results for 10 eigenvalues

10-Eigenvalue Placement Problem (10 runs)				
	Basic set		DSMP set	
Eigenvalue set	Dist error	Tree size	Dist error	Tree size
{-20±1j, -1±20j, -7±7j, -12±4j, -4±12j}	0.210	564.9	0.064 (30%)	385.6
{-1, -2, -3, -4, -5, -6, -7, -8, -9, -10}	0.267	564.5	0.023 (9%)	425.8

Results for a 16-eigenvalue assignment problem – a much more difficult problem – are shown in Table 5. The results for a complex conjugate 16-eigenvalue set {-20±1j, -1±20j, -7±7j, -12±4j, -4±12j, -15±2j, -9±5j, -5±9j} show that the average error of the basic set (0.279) is twice as large as that of the DSMP set (0.132). Mean size of the GP tree, BG size, and computation time are also given in Table 5. BG size represents the mean number of junctions and C, I, R elements in each individual. All mean tree sizes, BG sizes, and computation times of the DSMP set are less, respectively, than their basic set counterparts

Table 5. Results for 16 eigenvalues

16-Eigenvalue Placement Problem (10 runs) {-20±1j, -1±20j, -7±7j, -12±4j, -4±12j, -15±2j, -9±5j, -5±9j}							
Basic set				DSMP set			
Dist error	Mean Tree Size	BG Size	Compu. Time (min)	Dist error	Mean Tree Size	BG Size	Compu. Time (min)
0.279	663.1	62.2	72.4	0.132 (47%)	592.6	37	56.1

Although the experiments run to date are not sufficient to allow making strong statistical assertions, it appears that the search capability of the DSMP set is superior to that of the basic set for bond graph design. The superiority of the DSMP set seems very clear. Although the difference may be not seem large, it is very significant considering that the results of the basic set runs are already taking advantage of HFC (Hierarchical Fair Competition, Hu, *et al*., [9]).

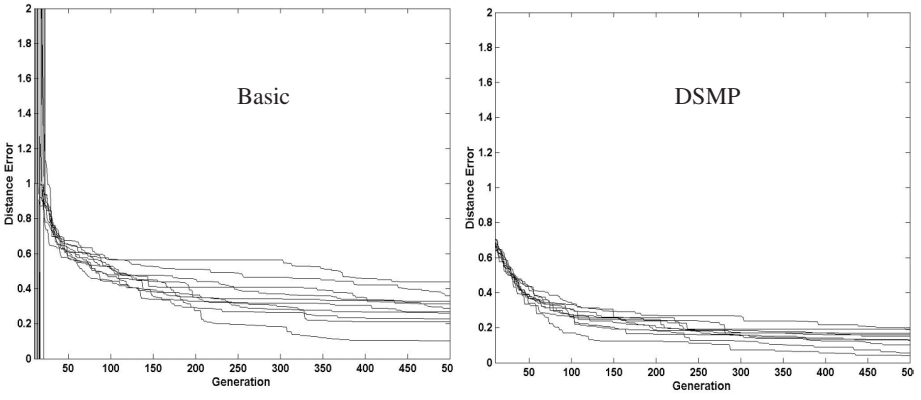

Fig. 6. Distance error for 16 eigenvalues

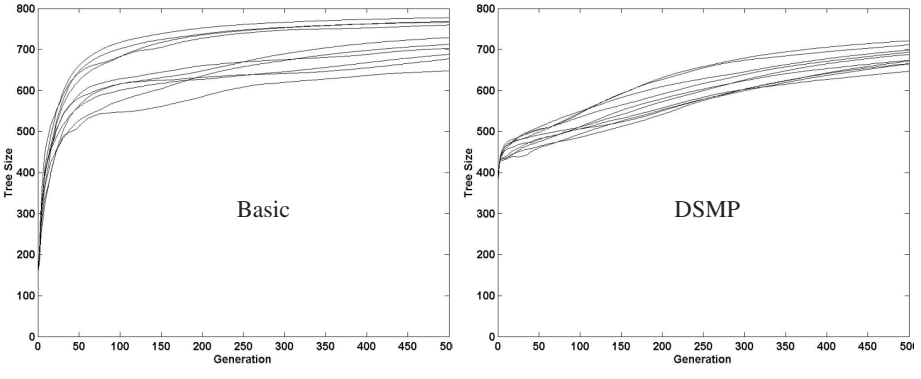

Fig. 7. Mean tree size for 16 eigenvalues

The distance errors (vs. generation) in 10 runs of the 16-eigenvalue problem are shown in Fig. 6. The distance errors of the DSMP set in Fig. 6 have already decreased rapidly within 50 generations, because only causally feasible (well-posed) individuals appear in the population. Figure 7 gives the mean tree sizes for each approach on the

16-eigenvalue problem. The DSMP set clearly obtains better performance using smaller trees. This bodes well for the scalability of the approach.

6 Conclusion

This paper has introduced the dense switched modular primitive for bond graph/GP-based automated design of multi-domain, lumped parameter dynamic systems. A careful combination is made of a *dense* representation (eliminating redundant components at junctions, guaranteeing causally well-posed bond graphs, and avoiding adjacent junctions of the same type) and a *switched* structure (allowing components that do not impact causal assignment at a junction to be present or absent depending on a binary switch). The use of these primitives considerably increases the efficiency of fitness assessment and the search performance in generation of bond graph models, to solve engineering problems with less computational effort.

As a proof of concept for this approach, the eigenvalue assignment problem, which is to synthesize bond graph models with minimum distance errors from prespecified target sets of eigenvalues, was used. Results showed better performance for various eigenvalue sets when the new primitives were used. This tends to support the conjecture that a carefully tailored, problem-specific representation and operators that generate only feasible solutions with smaller amounts of redundancy and fewer genotypes that map to the same effective phenotype will improve the efficiency of GP search. This, in turn, offers promise that much more complex multi-domain systems with more detailed performance specifications can be designed efficiently. Further study will aim at extension and refinement of the GP representations for the bond-graph/genetic programming design methodology, and at demonstration of its applicability to design of more complex systems.

Acknowledgment. The authors gratefully acknowledge the support of the National Science Foundation through grant DMII 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) 29–38
2. Koza, J.R., Bennett, F. H., Andre, D., Keane, M.A., Dunlap, F.: Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. IEEE Trans. Evolutionary Computation, vol. 1. no. 2. (1997) 109–128.
3. Koza, J.R., Bennett F.H., Andre D., Keane M.A., Genetic Programming III, Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, (1999)
4. Seo K., Goodman, E., Rosenberg, R.C.: First Steps toward Automated Design of Mechatronic Systems Using Bond Graphs and Genetic Programming. Proc. Genetic and Evolutionary Computation Conference, GECCO-2001, San Francisco (2001) 189

5. 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. Int. Jour. of Computers, Systems and Signals, vol. 3. no. 1. (2002) 55–70
6. Fan Z., Hu, J., Seo, K., Goodman E.D., Rosenberg R.C., Zhang, B.: Bond Graph Representation and GP for Automated Analog Filter Design. Genetic and Evolutionary Computation Conference Late-Breaking Papers, San Francisco (2001) 81–86
7. 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 (2002) 1073–1080.
8. Goodman E.D., Seo, K., Rosenberg R.C., Fan Z., Hu, J.: Automated Design of Mechatronic Systems: Novel Search Methods and Modular Primitives to Enable Real-World Applications. Proc. 2003 NSF Design, Service and Manufacturing Grantees and Research Conference, January, Birmingham, Alabama, (2003)
9. 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 (2002) 772–779.
10. Koza, J.R., Genetic Programming II: Automatic Discovery of Reusable Programs. The MIT Press, (1994)
11. Soule, T., Heckendorn, R.B.: Function Sets in Genetic Programming.: Proc. Genetic and Evolutionary Computation Conference, GECCO-2001, San Francisco (2001) 190
12. Karnopp, D.C., Rosenberg R.C., Margolis, D.L., System Dynamics, A Unified Approach, 3nd ed., John Wiley & Sons (2000)
13. Luke S., Strongly-Typed, Multithreaded C Genetic Programming Kernel. `http://www.cs.umd.edu/users/seanl/gp/patched-gp/.` (1997)
14. Zongker, D., Punch, W., lil-gp 1.1 User's Manual. Michigan State University, (1996)