Combining Bond-Graphs with Genetic Programming for Unified/Automated Design of Mechatronic or Multi Domain Dynamic Systems

S. A. Kayani[1], M. A. Malik[2]
Department of Mechanical Engineering,
College of Electrical and Mechanical Engineering,
National University of Sciences and Technology,
Peshawar Road, Rawalpindi-46000, Pakistan
{saheebk[1], mafzmlk[2]}@ceme.edu.pk

ABSTRACT
The multi domain nature of a mechatronic system makes it difficult to model using a single modeling technique over the whole system as varying sets of system variables are required. Bond-Graphs offer an advanced object oriented and polymorphic modeling and simulation technique. Bond-Graph model of the mechatronic system can be directly simulated on a digital computer using simulation softwares like 20-Sim© graphically or manipulated mathematically to yield state equations using a simplified set of power and energy variables. The simulation scheme can be augmented to synthesize designs for mechatronic systems employing genetic programming as a tool for open ended search. This research paper presents results of an experiment developed to combine Bond-Graphs with genetic programming for unified and automated design of mechatronic or multi domain dynamic systems.

Categories and Subject Descriptors
I.2.2 [**Artificial Intelligence**]: Automatic Programming; J.2 [**Physical Sciences and Engineering**]: Engineering.

General Terms
Design, Experimentation, Verification.

Key Words
Bond-Graphs, Genetic Programming, Unified/Automated Design, Multi Domain Dynamic or Mechatronic Systems.

1. INTRODUCTION
Traditionally a mechatronic system has been defined as a multi domain dynamic system combining mechanical, electrical, hydraulic, pneumatic and thermal components. To perform correctly mechatronic systems depend on the interaction of sensors, computers or microcontrollers and actuators. Taking complicated dynamic multi domain systems all the way from concept to prototype requires mathematical models. Such models can include those whose equations the modeler derives directly or develops with software that holds mathematics in the background. To model a mechatronic system all multi domain sub-systems must be connected and all non-linearities typical of a specific energy domain must be accounted for. To do this a language is needed to describe the different energy domains in communal terms. Using such a language sub-models can be connected in an overall system model which can then be simulated on a computer. Interacting physical systems store, transport and dissipate energy among sub-systems. Only Bond-Graphs can provide a concise pictorial representation of these interacting dynamic systems down to the topological level. [1] When using Bond-Graphs for mechatronic system representation we can assume that the mechatronic system under consideration is an n-port mechatronics network with e_1 and f_1 being system input effort and flow signals and e_n and f_n being system output effort and flow signals respectively. [2]

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

2. REVIEW OF UNIFIED/AUTOMATED DESIGN OF MECHATRONIC OR MULTI DOMAIN DYNAMIC SYSTEMS
The basic idea of unified and automated design is to replace the role of domain knowledge with the abundant computational resources available to the engineers these days. Genetic programming based simulated evolution techniques are capable of synthesizing designs of arbitrary complexity as the representation of designs is entirely open ended. [3]-[5] Figure 1 suggests a design approach for generating mechatronic systems identified by Jiachuan Wang et al. [2] in 2005. According to this methodology for any mechatronic system a start up design is specified at the initial stage. Then Bond-Graph representation is developed and it is transferred to the genetic programming tool which generates initial population, evaluates it according to the fitness function, reconfigures the population and repeats the process until the design criteria are met. The successful conceptual design candidates are transformed into final design again represented as Bond-Graph models. During this whole process information is extensively exchanged with knowledge caches and incorporated in initial and final stages of the design process. The methodology

involves increased human-computer interaction. Also with in the knowledge library stored information is updated and enhanced as the design proceeds. The knowledge library serves as a dynamic data base of domain knowledge which is constantly updated and the contents are verified continuously to remove any errors or omissions through an automated system. The input source to the knowledge library is the set of successful conceptual design candidates which means that only the best of all sorted information is stored for further reference and improvement.

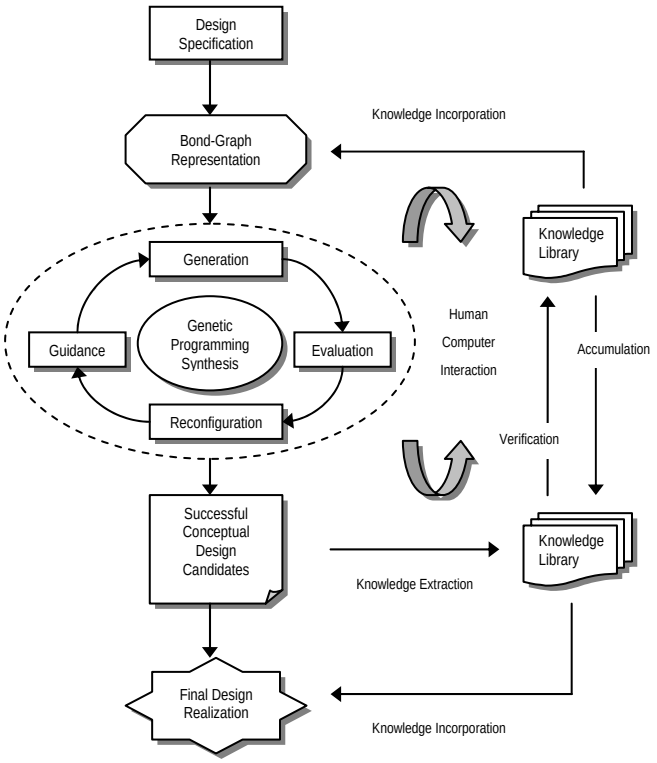

Figure 1. Automated design methodology for mechatronic or multi domain dynamic systems.

3. IMPLEMENTATION OF THE METHODOLOGY

This experiment is based on the methodology for unified/automated design of mechatronic systems identified by Kisung Seo et al. in [6] and Jianjun Hu in [7]. A brief summary of the methodology followed for implementing the automated design scheme is included. a. A Bond-Graph model is specified. b. First population of genetic programming trees is created. c. Each individual is evaluated for fitness using fitness function. d. Genetic programming operations i.e. selection, reproduction, crossover and mutation are performed for each population. e. Physical design is realized if termination condition of genetic programming run is satisfied. f. Otherwise the process is repeated starting from fitness evaluation of each individual. In [6] a two step process is employed for evaluation of Bond-Graph models. First each model is analyzed for causality and then state equations are derived identifying whether the system is linear or not. In the next step the fitness of the model is analyzed using fitness criterion. The experiment was designed to be as simple as possible because the C language code used for implementing the methodology became very complicated and difficult to handle. In all previous implementations of the methodology a UNIX based genetic programming software lil-gp 1.01 [8] has been used. For the first time Genetic Programming Studio 1.0 [9] is employed in this application which is based on lil-gp 1.01 kernel but offers a visual platform for executing genetic programming code. The code has been written using MS Visual C++ 6.0. Six different types of code files have to be developed. 1. protoapp.h contains prototypes of functions that app.c includes. 2. appdef.h contains #defines of the application. 3. app.h contains global data and any other function defined by user. 4. app.c contains software specific functions that help in input/output procedures. 5. function.h contains prototypes of functions and terminals of the problem. 6. function.c contains functions and terminals that are used for building the individual. Also files like epgdll.h, epgdll.c, defines.h, types.h, syscon.h and syscon.c are included for creating DLL or dynamic linked library files. These files are software kernel files and are not to be modified. The genetic programming parameters are saved in the problem set file with the extension .EPG. This software also offers a simulation tool which can be used for representing the individuals in LISP format. In case of Bond-Graph based individuals the final design requires simplification and reduction. The code is compiled and the DLL file is generated through MS Visual C++ 6.0. The problem implemented is an eigen value design problem from [6]. The two target eigen values selected are $-1\pm2j$ represented by cross marks × on complex plane and a Bond-Graph model with these eigen values is to be generated.

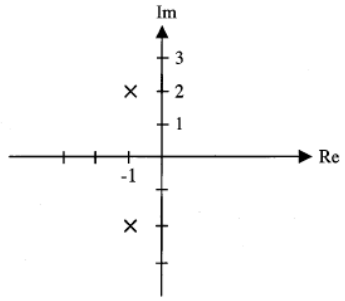

Figure 2. Representation of $-1\pm2j$ on complex plane.

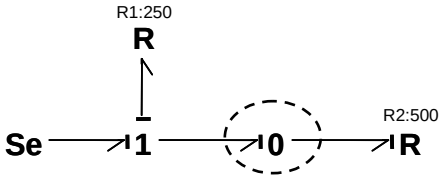

Figure 3. The embryo Bond-Graph model with modifiable site at the zero junction highlighted by dashed oval marking.

An embryo Bond-Graph model is specified with only one modifiable site highlighted by a dashed oval marking called the write head. The resistors are given same impedance values (R1:250, R2:500) as in [6] to achieve comparable results. The embryo is allowed only one modifiable site to keep the design

process and programming simple and less complicated to implement and interpret. The fitness function includes two parameters namely raw fitness and normalized fitness. Raw fitness $Fitness_{Raw}$ is the sum of distances between target eigen values and the nearest solution eigen values after they have been paired. Normalized fitness $Fitness_{Norm}$ is calculated according to the relation:

$$Fitness_{Norm} = 0.5 + \frac{1}{1 + Fitness_{Raw}} \qquad (1)$$

A list of genetic programming functions and terminals along with their descriptions appears in Table 1. The function f_add_R requires an additional parameter value called ephemeral random constant or ERC. An ERC is a special terminal whose value is fixed. When an ERC terminal is generated either during the filling of the initial population or by mutation later in the run, a value is attached to that terminal and is unchanged by subsequent operations. [8]

Table 1. Function and terminal descriptions

Function	Description
f_tree	Generate a tree model
f_add_C	Add a C element to a junction
f_add_R	Add a R element to a junction
f_add_I	Add an I element to a junction
f_insert_J0	Insert a zero junction in a bond
f_insert_J1	Insert a one junction in a bond
replace_C	Replace with C element
replace_R	Replace with R element
replace_I	Replace with I element
f_add_ERC	Add two ERCs
f_del_ERC	Delete two ERCs
end_A	End terminal for add element
end_I	End terminal for insert element
end_R	End terminal for replace element
ERC	Ephemeral Random Constant

The genetic programming parameters used in the experiment have been included in Table 2. The software was installed and run on a DELL/Pentium-III/1.0GHz and 256MB RAM personal computer with Windows XP/2002/SP-1.

Three different random seeds were used and the experiment was repeated three times with population sizes of 100, 1000 and 2500 with different number of generations. When different Bond-Graph functions can be applied to the same write head this technique is termed as strongly typed genetic programming.

Add functions can only be applied to a junction while insert functions are only applied to a bond. Replace functions change the type of the Bond-Graph element and are node specific. Arithmetic functions of addition and subtraction are carried out by

Table 2. Genetic programming parameters

Number of Generations	100-500
Population Size	100-2500
Initial Population	Half and Half
Sub Populations	10
Maximum Nodes	300
Initial Depth	3-6
Maximum Depth	17
Selection	Tournament
Size	7
Crossover	0.9
Mutation	0.1

f_add_ERC and f_del_ERC respectively. The developmental or construction procedure for a Bond-Graph model or phenotype is identified by the genetic programming tree or the genotype. Using LISP format of representing genetic programming trees the simulation tool of Genetic Programming Studio 1.0 is used to print long hand versions of such genetic programming trees which need to be simplified for extracting meaningful information. One genetic programming tree represents one individual.

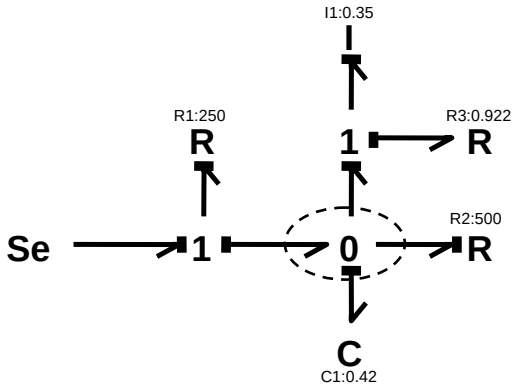

Figure 4. The final simplified Bond-Graph model.

The average distance error *e* between target and solution eigen values is calculated using distance formula for two pairs of numbers (x_1, y_1) and (x_2, y_2) given as:

$$e = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{½} \qquad (2)$$

The best solution eigen values compared to target eigen values are included in Table 3 along with average distance error. This table also contains number of R, C, I and junction elements added to the write head. Numerical values of one port elements are also shown. It is to be noted that the eigen values are determined using the A matrix of the Bond-Graph model (when state-space equations are written in matrix form assuming the system is linear) containing state variables contributed by the energy storing C and I elements.

$$\frac{dy}{dt} = y[A] + u[B] \qquad (3)$$

In above equation y is the vector of states, A is an $n \times n$ square matrix, u is the array of sources and B is a matrix of dimensions $n \times m$ where n is the number of states and m is the number of sources.

Table 3. Summary of results

Target Eigen Values	
$-1 \pm 2j$	
Solution Eigen Values	
$-0.78 \pm 1.063j$	
Average Distance Error	
0.961	
Evolved Structure on Write Head	
R Elements	1
C Elements	1
I Elements	1
Junctions	1
Bonds	4
Bond-Graph Element Values	
R Element	0.922
C Element	0.42
I Element	0.35

The values achieved in this experiment are slightly different from the values in [6]. Limitations in writing the code used for generating the Bond-Graph models and tuning of the fitness evaluation process may be the reason which can be removed with more rigorous effort devoted towards problem implementation.

4. FINAL COMMENTS AND CONCLUSION

The methodology followed has been proposed for unified and automated design of mechatronic or multi domain dynamic systems using Bond-Graphs for system representation and genetic programming for exploring the design space in an open ended manner.

This research paper has been a product of an indigenous attempt to implement the design methodology for a postgraduate level research project. The objective was to repeat and/or develop a simple experiment based on the said methodology and implementation scheme to achieve comparable results. As the C language code was developed without referring to any source so certain limitations were unavoidable resulting in final eigen values falling short of the target. However the implementation and results prove that the methodology is valid and thus verified. The robustness of the approach lies in compactness of the genetic programming code and fitness evaluation of the evolved designs. The interpretation of the results will be simplified in further research efforts. The complexity of the implementation especially code development and final elucidation is one of the reasons that this research area remains relatively less explored. So far results of only one research group have been published. [2][6][7] The problem implemented in this research paper though simple gives perhaps the first independent verification of the design methodology identified by this particular research group. It is intended that this methodology will be followed for extending the automated design concept to more sophisticated mechatronic systems like kinematic sections of humanoid robots, end effectors and even synthesis of a simple two legged walking robot.

5. ACKNOWLEDGMENTS

Mr. Ali Hassan/DCE and Mr. Wasim Baig/OIC Library deserve our regards for their contributions over a period of time towards exploring available literature on Bond-Graph modeling and simulation and genetic programming. This research was carried out as part of a postgraduate dissertation in College of E&ME, National University of Sciences and Technology, Rawalpindi, Pakistan. The help of the institution is appreciated.

6. REFERENCES

[1] Karnopp, D. C. and Margolis, D. L. The Language of Interaction. *ME Magazine*, American Society of Mechanical Engineers, January 2001, 1-4.

[2] Wang, J., Fan, Z., Terpenny, J. P., and Goodman, E. D. Knowledge Interaction with Genetic Programming in Mechatronic Systems Design using Bond-Graphs. *IEEE Transactions on Systems, Man and Cybernetics*, Part C: *Applications and Reviews*, Vol. 35, No. 2, (May 2005), 172-182.

[3] Koza, J. R., Keane, M. A., and Streeter, M. J. What's AI Done for Me Lately? - Genetic Programming's Human Competitive Results. *IEEE Intelligent Systems*, 18(3), (2003), 25-31.

[4] Hirsh, H., Banzhaf, W., Koza, J. R., Ryan, C., Spector, L., and Jacob, C. Genetic Programming. *IEEE Intelligent Systems*, 15(3), (2000), 74-84.

[5] Koza, J. R., Bennett, F. H., Lohn, J., Dunlap, F., Keane, M. A., and Andre, D. Automated Synthesis of Computational Circuits using Genetic Programming. In *Proceedings of IEEE International Conference on Evolutionary Computation*, Indianapolis, 1997, 447-452.

[6] Seo, K., Hu, J., Fan, Z., Goodman, E. D., and 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, (2002), 55-70.

[7] Hu, J. *Sustainable Evolutionary Algorithms and Scalable Evolutionary Synthesis of Dynamic Systems*. PhD Dissertation, Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, 2004, 69-80.

[8] Zongker, D. and Punch, W. *lil-gp 1.01 User Manual*. Michigan State University, East Lansing, MI, USA, 1996.

[9] Novales, A. C. *Genetic Programming Studio 1.0 User Manual*. University of Cordoba, Spain, 1998.