Evolutionary Design of Both Topologies and Parameters of a Hybrid Dynamical System

Jean-François Dupuis, Zhun Fan, *Senior Member, IEEE,* and Erik D. Goodman

***Abstract*—This paper investigates the issue of evolutionary design of open-ended plants for hybrid dynamical systems, i.e., both their topologies and parameters. Hybrid bond graphs (HBGs) are used to represent dynamical systems involving both continuous and discrete system dynamics. Genetic programming, with some special mechanisms incorporated, is used as a search tool to explore the open-ended design space of hybrid bond graphs. Combination of these two tools, i.e., HBGs and genetic programming, leads to an approach called HBGGP that can automatically generate viable design candidates of hybrid dynamical systems that fulfill predefined design specifications. A comprehensive investigation of a case study of DC-DC converter design demonstrates the feasibility and effectiveness of the HBGGP approach. Important characteristics of the approach are also discussed, with some future research directions pointed out.**

***Index Terms*—Automated design, bond graphs, evolutionary design, genetic programming, hybrid mechatronic systems.**

I. Introduction

A LINE OF RESEARCH has been pursued to automate the design process of mechatronic systems [1]–[5]. This paper presents research that aims to extend work on evolutionary design of mechatronic systems based on the bond graph by genetic programming (BGGP) approach that combines bond graphs (BG) and genetic programming (GP) [6]. Interesting results on a variety of case studies were obtained using the BGGP approach, including electrical filter design [7], [8], typewriter redesign [9], design of mechanical vibration absorbers [10], embodiment of a vehicle suspension system via co-design of controller and plant [11], [12], and micro-electro-mechanical systems [13]. However, all of them are limited to dealing only with continuous dynamics of mechatronic systems. In addition, even though bond graphs can also be used to represent controllers, such as proportional integral-derivative or phase-lead controllers for continuous dynamics of a hybrid mechatronic system [12], they do not have the capacity to describe controllers that govern the discrete parts of the hybrid mechatronic system. The proposed extension makes use of hybrid bond graphs (HBGs) to represent both continuous and discrete dynamics, and GP to explore the open-ended design space of hybrid mechatronic systems, leading to the HBGGP approach. With this method, systems containing both event-driven and time-driven phenomena can now be evolved using GP and bond graphs, considerably increasing the potential applications previously covered with the BGGP approach. The complexity of the methodology is, however, increased by the need of controlling discrete events. A lookahead controller was adopted to fulfill this task.

Actually, various representations, including a finite state automaton (FSA) controller and a lookahead controller, have been investigated in the previous work in an effort to identify the best representations for hybrid controllers [14]–[16]. FSA controllers have gained popularity in industry, and appeared to be an effective representation. It has also been shown that proper FSA controllers can be designed automatically using an evolutionary approach. The controllers obtained were able to successfully control a given hybrid mechatronic system to achieve both predefined and varying control targets [16]. However, the evolutionary design of FSA controllers in this way appeared to be too computationally expensive, which interferes with the possibility of concurrently designing both controller and plant of a hybrid mechatronic system, and thus reduces the robustness of exploration for open-ended plant designs. In fact, when we tried to evolve both the controller represented by FSA and the plant represented by hybrid bond graph, we found it impossible to obtain any reasonable results in an acceptable period of time, because the search space became prohibitively large. As an alternative, we have adopted the lookahead controller to take advantage of its forthright design procedure that can be easily automated. Therefore, for each individual design candidate of the plant, an optimized lookahead controller can be obtained automatically with a fixed procedure. Our algorithm can then focus on exploration of the open-ended design space of plants, with the definition of an optimized lookahead controller and the evaluation of the system performance being automated thereafter.

This paper makes three primary contributions.

1) The HBGGP method uses genetic programming to generate hybrid bond graphs so that it can also deal with dynamical systems that contain discrete events.

Manuscript received December 6, 2010; revised March 17, 2011; accepted May 4, 2011. Date of current version May 24, 2012. This work was supported in part by the Danish Agency for Science Technology and Innovation, under Grant 645-06-0186, in part by the National Science Foundation under Cooperative Agreement DBI-0939454, and in part by the National Nature Science Foundation of China under Grant 61175073. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

J.-F. Dupuis is with the Department of Management Engineering, Technical University of Denmark, Lyngby 2800, Denmark (e-mail: jedu@man.dtu.dk).

Z. Fan is with the College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China (e-mail: zfan@tongji.edu.cn).

E. D. Goodman is with the BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48824 USA (e-mail: goodman@msu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2011.2159724

TABLE I
POWER VARIABLES IN VARIOUS ENERGY DOMAINS

Energy Domain	Effort	Flow Variables
Mechanical translation	Force	Linear velocity
Mechanical rotation	Torque	Angular velocity
Electrical	Electromotive force	Current
Magnetic	Magnetomotive force	Flux rate
Hydraulic	Pressure	Volumetric flow rate
Thermal	Temperature	Entropy flow rate

2) The lookahead controller is adopted as a representation for the discrete controller so that the search space of the hybrid system design is effectively reduced.
3) A careful design of primitives and speciation mechanisms applied to genetic programming ensures efficient exploration of the reduced search space, and enables the algorithm to find valid design candidates satisfying predefined design specifications.

The remainder of this paper is organized as follows. First, Section II introduces hybrid bond graphs and the lookahead controller. Then, Section III describes the HBGGP methodology that automatically generates viable designs of hybrid dynamical systems represented by hybrid bond graphs. Section IV provides a case study of design of a DC-DC converter circuit. Finally, Section V concludes this paper with observations and discussions of future research directions.

II. HYBRID BOND GRAPHS AND LOOKAHEAD CONTROLLERS

A. Hybrid Bond Graphs

Bond graphs [17], [18] and their hybrid extension [19], [20] are graphical representations of the energy exchange in a physical system. Bonds between components model the transfer of power between them. In a bond, the power is divided into its two subcomponents, effort and flow, whose product is power. Using such a power-based modeling approach makes it very easy to span multiple domains in a single representation. Table I lists some effort and flow equivalents for various physical domains.

The components used in a bond graph model are also very generic. They simply model how energy is generated (S_e, S_f), stored (C, I), transformed (GY, TF), or dissipated (R). These components can be connected through junctions at which they share either the same effort value (0-junction), or flow value (1-junction). Tables II–IV summarize the components used in a bond graph model.

In order to model discrete phenomena within physical system, a switch component (Sw) was added to the set of bond graph components to allow the creation of hybrid bond graphs. The switch acts as a zero-effort or zero-flow source depending on its state. For example, if a switch is connected to a 1-junction, the zero-flow state of the switch will block any flow through that junction. On the other hand, the zero-effort state will add a zero element in the effort summation, hence correctly modeling an ideal switch.

TABLE II
ENERGY GENERATION, STORAGE AND DISSIPATION COMPONENTS

Symbol	General Relation	Linear Relation	Example
$\rightharpoonup R$	$e = \Phi_R(f)$	$e = Rf$	Damping, friction, restriction
	$f = \Phi_R^{-1}(e)$	$f = e/R$	
$\rightharpoonup C$	$q = \Phi_C(e)$	$q = Ce$	Spring, tank, capacitor
	$e = \Phi_C^{-1}(q)$	$e = q/C$	
$\rightharpoonup I$	$p = \Phi_I(f)$	$p = If$	Mass, coil, inertia
	$f = \Phi_I^{-1}(p)$	$f = p/I$	
$S_e \rightharpoonup$	$e = \Phi_e(t)$		Gravity, voltage source
$S_f \rightharpoonup$	$f = \Phi_f(t)$		Pump, current source

TABLE III
ENERGY TRANSFORMATION COMPONENTS

Symbol	General Relation	Example
$\underset{f_1}{\overset{e_1}{\rightharpoonup}} TF \underset{f_2}{\overset{e_2}{\rightharpoonup}}$	$e_1 = me_2$ $f_2 = mf_1$	Lever, pulley, gears
$\underset{f_1}{\overset{e_1}{\rightharpoonup}} GY \underset{f_2}{\overset{e_2}{\rightharpoonup}}$	$e_1 = rf_2$ $e_2 = rf_1$	DC motor, hall effect sensor

TABLE IV
POWER JUNCTIONS

Symbol	General Relation
$\begin{array}{c} \quad e_3 \downarrow f_3 \\ \underset{f_1}{\overset{e_1}{\rightharpoonup}} 0 \underset{f_2}{\overset{e_2}{\leftharpoonup}} \end{array}$	$e_1 = e_2 = e_3$ $f_1 + f_2 + f_3 = 0$
$\begin{array}{c} \quad e_3 \downarrow f_3 \\ \underset{f_1}{\overset{e_1}{\rightharpoonup}} 1 \underset{f_2}{\overset{e_2}{\leftharpoonup}} \end{array}$	$f_1 = f_2 = f_3$ $e_1 + e_2 + e_3 = 0$

B. Lookahead Controller

A one-step lookahead controller was implemented to control an example DC-DC converter. The one-step lookahead controller was chosen mainly due to its simplicity. This controller will, at run time, for each time step, perform a simulation in order to predict the future state variable values of the system, x_{Sw_i}, for all switch configurations, $SW = \{Sw_i | i = 1, \ldots, n\}$. Then the chosen configuration to be applied at the next time step will be the one having the direction closest to that of the target point x_d from the current state. Thereby, the controller minimizes the angle between the vector defined by the current state x and the target x_d, and the vector defined by the current state and the predicted system trajectory x_{Sw_i}

$$\underset{Sw_i \in SW}{\operatorname{argmin}} \left(\arccos \frac{(x - x_d)(x - x_{Sw_i})}{|x - x_d||x - x_{Sw_i}|} \right). \tag{1}$$

An important characteristic of this type of controller is that the controller does not need to be trained or redesigned for a specific system. The controller only needs an accurate model of the system to be controlled in order to be able to predict the future state of the system. The only parameter of the controller

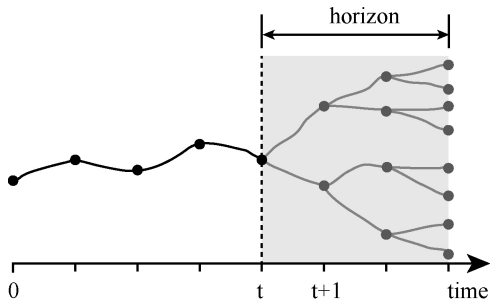


Fig. 1. Lookahead control with limited horizon.

is the lookahead horizon, which should be established with regards to the available computing power and sampling time between two adjacent controller commands. In our study, the lookahead horizon was set to a single step in order to minimize computation time.

III. Evolving Hybrid Bond Graph Using Genetic Programming

GP has been applied in the past to a wide variety of domains. In the beginning, GP was intended to evolve computer programs [21]. Soon, people wanted to exploit the power of GP to generate things other than tree-like systems. In [22], the cellular encoding is proposed to evolve neural networks, in which programs are interpreted as a sequence of instructions that modify (grow) a simple initial structure (embryo). This approach has notably been extended to evolve electronic circuits [23] and to numerous other domains. The edge encoding [24] has also been proposed as an alternative to the cellular encoding to evolve general graphs. This approach is very similar to the cellular encoding, but it uses edge operators rather than node operators. A variation of the edge encoding [25] has also been applied to evolved growing creatures based on L-systems [26].

Following an idea resembling the cellular encoding, the BGGP [6], [7] approach was proposed. Interesting results on a variety of case studies were obtained, including electrical filter design [7], [8], typewriter redesign [9], design of mechanical vibration absorbers [10], co-design of controller and plant embodiment of a vehicle suspension system [11], [12], and design of micro-electromechanical systems [13].

As with cellular and edge encodings, an embryo is first defined. Then a tree is generated by GP, containing bond-graph-modifying functions that are executed to transform the embryo into the bond graph to be evaluated. Unlike the previous approaches, the BGGP operators can be applied on both nodes (components) and edges (bonds) of the bond graph.

In order to evaluate an individual, its GP tree must first be executed to grow the bond graph from the common embryo. The growing process usually leads to a bond graph containing some junctions and components that can be simplified. Therefore, a bond graph simplification algorithm is applied before formulating the state-space equation of this bond graph. Once the simplified hybrid bond graph is obtained, the number of switches it contains is passed to the lookahead controller

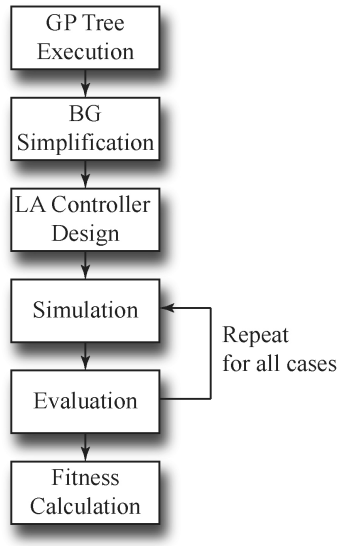


Fig. 2. Individual evaluation procedure.

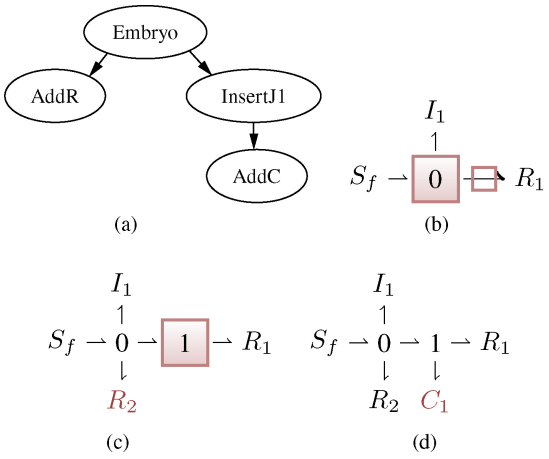


Fig. 3. Simple example of bond graph creation from an embryo. (a) GP tree. (b) Embryo. (c) After AddR and InsertJ1. (d) After AddC.

before the integration of the state-space equation takes place. The system can be simulated for many different scenarios. The performances obtained for each of them are combined to form the fitness that will be returned to the evolver. Fig. 2 summarizes this evaluation procedure.

A. Introduction to Bond Graph Construction

From the standard GP point of view, the GP tree used here to grow hybrid bond graphs could be interpreted up-side down, as the primitives are defined to receive a single argument and to return a variable number of them. The received argument is the modification point where the primitive will be applied. After the execution of the primitives, a list of new modification points where the bond graph can be further modified is returned.

Fig. 3 shows a very simple example of how a bond graph can be created from a GP tree.

It starts with an embryo, which is normally composed of two parts: the subsystem that is not changeable, and modifiable

sites. The modifiable sites are carefully defined in the embryo so that new structures of the system can be grown from them. An embryo can be either small or big. In the former case, almost the whole structure of the system needs to be grown out of the embryo. In this case, the designer usually has no or little knowledge on what the resultant design could be. In the latter case, the major part of the design is usually known in advance, and a certain subsystem needs to be redesigned. In this case, it is worthwhile to note that previously obtained solution can also be used as an embryo, so that successive evolutionary design can be conducted [7], [9]. Here, the embryo indicates that the system input will be a flow source. The resistance, R_1, could model, in the electronic domain, an output load to which the system will be connected. Also, there is a known amount of inertia that will be present in the system, represented by including I_1 in the embryo. The embryo also defines the modification sites where the bond graph will be allowed to grow. Here, the 0-junction represents a junction modification point, while the bond connecting R_1 represents a bond modification point.

In the illustration, starting from the embryo, a primitive, which may include new modification sites, is connected to each modification site. Here, the AddR primitive is connected to the 0-junction site and the InsertJ1 is connected to the bond modification site. InsertJ1 will make the newly inserted junction a new modification site that will be used by the AddC primitive. Fig. 3(d) shows the resulting bond graph structure obtained after the complete execution of the GP tree represented by Fig. 3(a) on an embryo represented by Fig. 3(b).

B. Typed GP for Bond Graph Evolution

In the BGGP approach, bond graphs are modified at specific modification points. These points can be located on a bond or a component, which in turn can be a junction or a terminal component like a resistor or a capacitor. In order to have a relevant operator applied at a certain modification point, the GP tree needs to be constrained to only allow connections between compatible GP primitives.

The initial BGGP method was extended to use a variation of the usual strongly typed GP structure. Instead of having a single argument type, each GP primitive defines a set of them. Each returned argument could also define a set of valid types, but the primitives currently used here only return a single type per value.

The main advantage of including a set of supported input argument types is to be able to introduce more flexibility during the modification of the GP tree. For example, two primitives could modify the same junction in the bond graph. The first one could be valid only if a 0-junction is given, while the second one could be valid on either junction type. Therefore, the set of the input argument types of the first primitive will only include the 0-junction type, while the set of the second one will include both junction types. As a result, both primitives will be able to be matched at a 0-junction modification point. However, only the second primitive could be attached to a 1-junction site. As a result, the possible crossover and mutation points for the second primitive are greatly enhanced.

TABLE V
DEFINITION OF ARGUMENT TYPES

Type	Description
Bond	Site located on a bond.
1-Bond	Site located on a bond connecting a 1-junction.
0-Bond	Site located on a bond connecting a 0-junction.
Jct	Site located on a junction.
1-Jct	Site located on a 1-junction.
0-Jct	Site located on a 0-junction.
Node	Site located on a terminal bond graph component.
Double	Parameter value of a bond graph component.

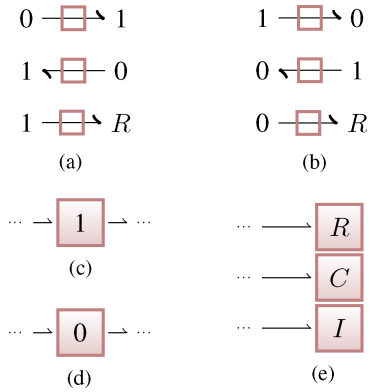


Fig. 4. Representation of bond graph argument types. (a) 1-Bond. (b) 0-Bond. (c) 1-Jet. (d) 0-Jct. (e) Node.

Without the use of a set to define the valid input argument types, multiple versions of the same primitive would be required to cover all cases. It would therefore prevent the crossover and mutation operators from changing the type of modification site on which the primitive could be applied. For instance, if two versions of the same primitive were defined to cope with two different cases, then when both of them appeared in the GP tree, it would not be possible for the crossover operator to exchange those sub-trees as their types would not match. On the other hand, if a single instance of the primitive is defined with a set of valid argument types, these two sub-trees can be swapped by the crossover operator. Hence, the genetic operators can more easily transport good building blocks among different locations in the GP tree.

C. Bond Graph Argument Types

The argument types used in the proposed approach are summarized in Table V and their graphical representation are illustrated in Fig. 4. There are four main argument types–one for each category of modification site.

The first category represents the modification sites that are located on a bond. It can be a bond between two junctions or between a junction and a terminal component. In the former case, if the power stroke is located on the side of a 1-junction, the bond will be called a 1-Bond; otherwise, it will be called a 0-Bond. In the latter case, the type of bond will depend on the junction type to which it is connected. The argument type "Bond" serves as a wild card that can mean any bond type.

The second category consists of the modification sites that are located on a junction. Depending on the junction type, the argument type will be called a 1-Jct or a 0-Jct to refer to a modification site located on a 1-junction or on a 0-junction,

respectively. Again, the argument type "Jct" defines a wild card that means any junction type.

The other two categories are related to terminal components. The Node argument type refers to a modification site that is located on a terminal component such as *R*, *C*, *I*, or *Sw*. The Double argument refers to a real number value, which is mainly used to establish the parameter value of the related component.

Finally, there is a special argument type called Parent's. This return argument type will take the same type as the one received as input. This is useful for a primitive that can be applied to a variety of modification site types, but needs to return a corresponding type for its child primitives.

D. Primitives

In order to successfully evolve a bond graph, a set of primitives that alter the current topology of the bond graph needs to be defined. Table VI lists the primitives used in the proposed approach. Most of them are a redesign or an extension of the ones used in the original BGGP approach.

These primitives are designed to be applied on a single modification site. Then the newly created modification sites, as well as the site received as input, are returned as arguments, so that they can be further modified by later primitives.

1) *Closing Modification Site:* In order to stop the growth of a bond graph at a specific location, and thus have a GP tree of finite size, a terminal primitive is needed. This primitive is simply called End. It is used to close a modification site so that further modification is not possible. It, together with the ephemeral double (E) forms the terminal set of primitives.

2) *Component Related Primitives:* The add-component primitive connects a new component to an existing junction. The primitive exists in the form of AddR, AddC and AddI, which respectively add a resistor-, a capacitor-, and an inductor-like component. This primitive acts on both junction types and returns the junction type on which it was applied.

In addition, this primitive introduces a root point of a parameter sub-tree. This sub-tree forms a real number expression that computes the parameter value of the added component. It also introduces a node modification site, where the replace component primitive can be applied. This latter one is used to convert an existing component to a new one with its own parameter sub-tree.

An important component-related primitive is the switch-addition primitive. The addition of switches to the bond graph makes the emergence of hybrid systems possible. It should be noted that the number of switches in a hybrid bond graph should be constrained within a certain acceptable range. This constraint ensures that the control complexity of the system does not explode. For instance, the simulation time of the system using a lookahead controller will increase exponentially with the number of switches. It is therefore advisable to limit the number of switches present in a hybrid bond graph to avoid excessive evaluation time.

3) *Junction Related Primitives:* Initially, only the simple insertion primitive was used in its two forms, InsertJ0 and InsertJ1, which insert a 0-junction and a 1-junction, respectively. These primitives insert new junctions of the specified types

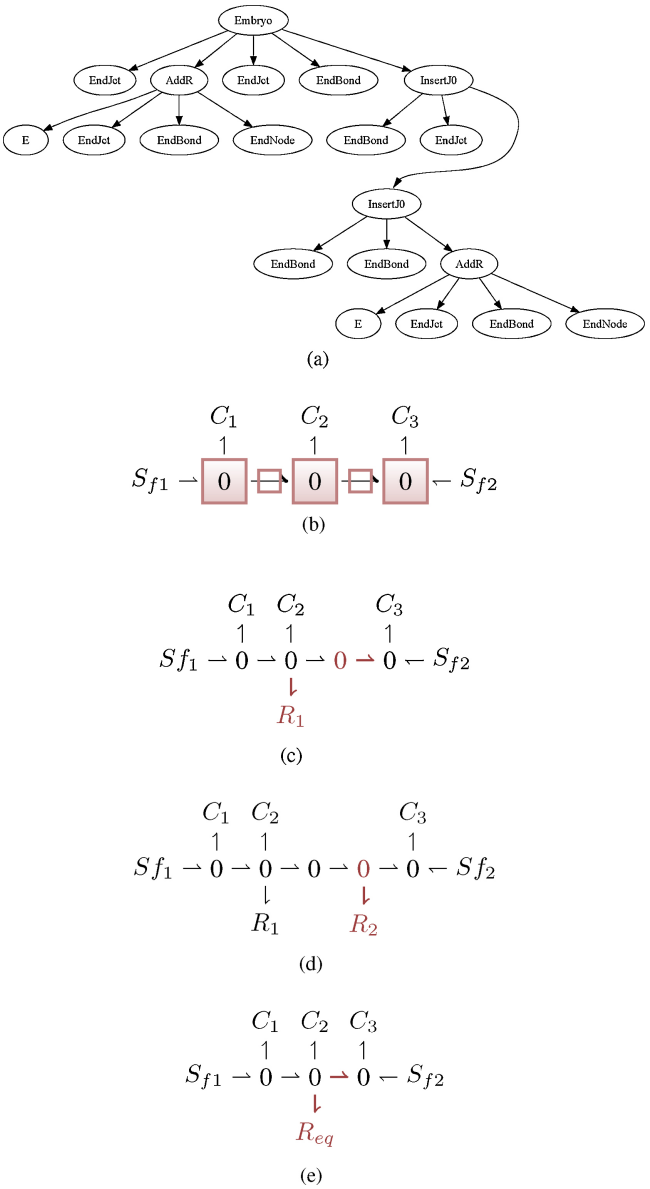


Fig. 5. Example of ineffective primitive action on bond graph. (a) GP tree. (b) Embryo of a three-tank system with two inputs. (c) After AddR and InsertJ10. (d) After InsertJ01 and following AddR. (e) After simplification.

regardless of the junctions to which they will be connected. Because of the properties of adjacent junctions of the same type, this means that many redundant junctions can be inserted successively, which will later be simplified into a single one. As a result, the GP tree can grow to a substantial size, but the net effect of this growth on the simplified bond graph can still be negligible. It is worth noting that many applications of these primitives have no net effect on the resulting bond graph, but that these negligible operations are easily detected and removed before state equations are formulated.

Fig. 5 illustrates the problem of ineffective junction insertion. Both 0-junction insertion and the following resistance addition operations vanish after simplification, as both resistances are combined. Therefore, parts of the GP tree could be removed to leave only the AddR primitive with a new parameter computation sub-tree that reflects the merger of the two resistances.

As GP already suffers from the bloating effect [27], it would seem be worthwhile to design primitives that do not stimulate bloating. For this reason, the junction pair insertion primitive was designed. Instead of inserting a single junction, this primitive inserts a connected pair of 0 and 1-junctions. The order of the two junctions assures that two junctions with the same type are not connected to each other. This tends to reduce the number of redundant junction insertions.

While having the potential to reduce bloating, the junction pair insertion does not eliminate the possibility of topologically ineffective sub-trees. For instance, if one of the two inserted junctions is not later connected to another component, it results in an empty junction which will be eliminated during the simplification phase. As a result, two similar junctions will now be connected together and further simplifications will be possible. In fact, the simplification step usually reduces the size of the evolved bond graph by 60%.

In the end, the insert junction pair can be seen as a way to stimulate the use of alternating 0 and 1-junction in the creation of the bond graph, while not completely removing potentially valuable introns in the genotype.

4) *Loop Creation Primitive:* The SplitBond primitive is used to introduce loops in the system by splitting the power into two parallel paths. Providing a loop generation primitive in the GP tool set greatly improves the variety of systems the evolutionary search can find.

5) *Flip Bond Primitive:* The flip bond primitive is used to invert the power direction of a bond. It can be used to control the power direction in the bond graph, notably in loops. This primitive is similar to the primitive that inverts the polarity of a component when evolving electrical circuits [28].

E. Genetic Operators

The genetic operators applied on the GP tree to evolve hybrid bond graphs are essentially the same as the ones used in standard GP. However, the crossover and mutation operators used for basic strongly typed GP have been modified to balance topological exploration and parameter optimization. In addition, the operator set is augmented by the speciation operator.

1) *Crossover and Mutation:* In this paper, we adopted a crossover rate of 0.9 and mutation rate of 0.1. In standard GP, crossover and mutation operators randomly select the nodes where the tree will be modified. All nodes thus have an equal chance of being selected. Since the parameters are defined by numerical sub-trees within the same tree as the structure nodes, parameter and structure nodes are chosen with the same probability regardless of their nature. Because there are often more numerical nodes than structure-modifying ones, structure exploration is slowed down as more parameter-modifying nodes are generated and are therefore more likely to be picked by the crossover or mutation operators. As a result, premature structural convergence can occur, as only parameter nodes are likely to be selected for optimization.

In order to avoid this problem, structure and parameter exploration must be controlled explicitly. Reference [29] proposed to use a probabilistic method to first decide if a parameter or structure modification will be made. Then, mutation or crossover are applied only to the relevant type of nodes. Since new structures need time to tune their parameters, a bias is introduced toward numerical nodes. As a result, newly obtained structures have the opportunity to adjust their parameters in order to show their full potential before being discarded. A typical parameter versus structure modification rate is as follows:

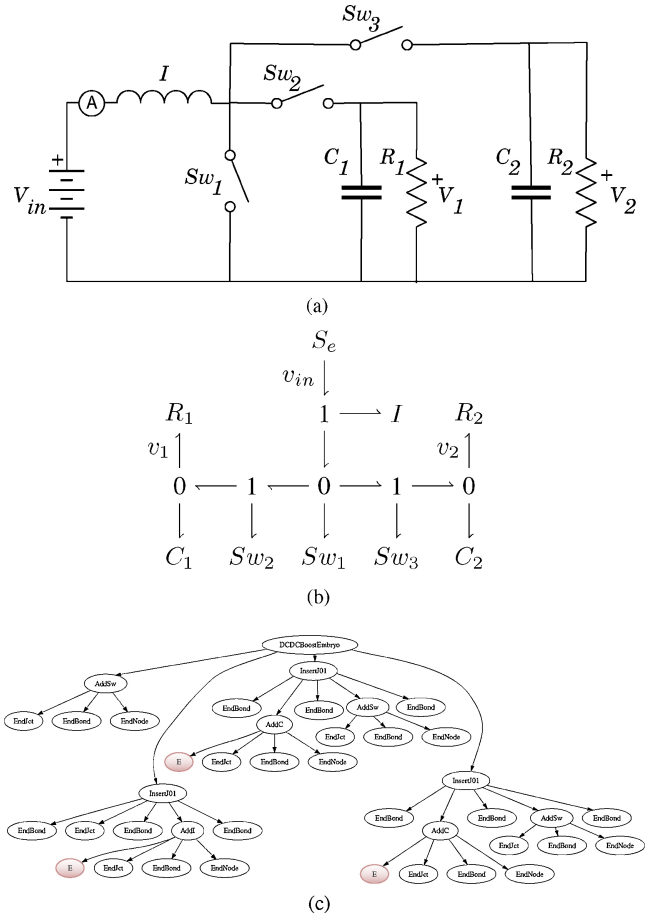


Fig. 6. Hand-designed single-input double-output DC-DC converter. (a) Physical system schematic. (b) Bond graph. (c) (GP-Tree).

$$p_{\text{struct}} = 0.15$$
$$p_{\text{param}} = 0.85.$$

Reference [29] also proposed to control explicitly the process of node selection when applying a parameter modification in order to achieve balanced parameter evolution. First, the root of each numerical sub-tree is listed and one of them is chosen randomly with equal probability. Then, mutation or crossover is applied to the sub-tree of the chosen root. This process gives equal mutation and crossover probabilities to each parameter as it avoids focusing attention on an oversized numerical sub-tree.

2) *Speciation:* We propose to use several speciation mechanisms to prevent premature convergence of genetic programming. These mechanisms include structure fitness sharing (SFS), hierarchical fair competition (HFC), and utilization of an aging factor. Premature convergence in evolutionary computation has always been an important issue. Many ap-

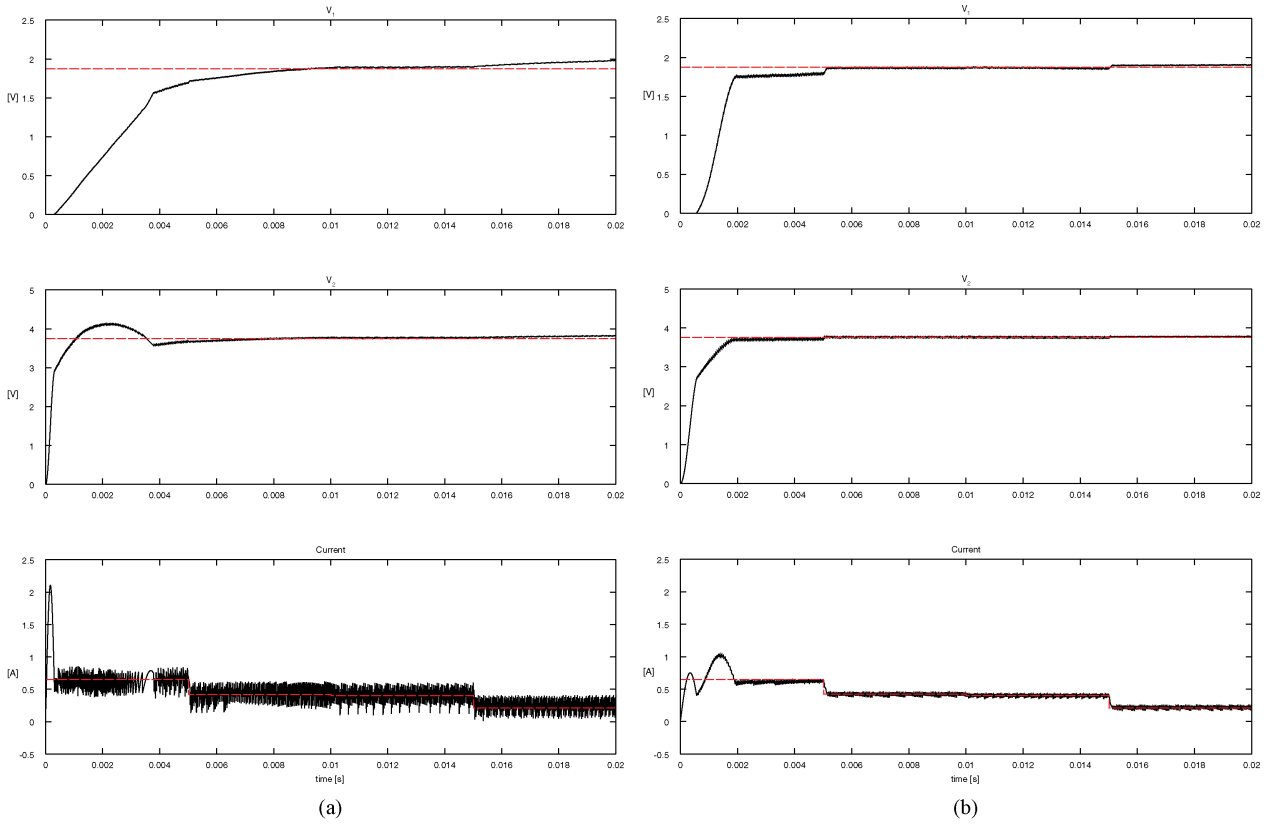


Fig. 7. Performance of the hand-designed single-input double-output DC-DC converter shown in Fig. 6(a). (a) Initial values. (b) Optimized values.

proaches have been proposed throughout the years to keep a good balance between exploration and exploitation. In genetic algorithms (GA), common diversity maintenance techniques include crowding [30], deterministic crowding [31], and fitness sharing [32]. Those techniques work quite well for solving problems that involve a fixed number of parameters.

On the other hand, when facing problems where only the structure needs to be evolved, other methods have been proposed. One approach is to use a multiobjective fitness that includes a distance metric between two GP trees to enforce diversity [33], [34]. A second method, implicit fitness sharing [35], proposes to look at the results produced by the individuals. In this approach, individuals are grouped together based on their behaviors (or phenotypes) instead of their genotypes.

However, when dealing with problems that involve the optimization of both structure and parameters, the landscape of the search space is changed significantly. Structure diversity is required in order to discover innovative designs, but the performance of a given structure can only be assessed if enough parameter optimization has been done on that structure. Therefore, a balance between structure and parameter optimization must be achieved.

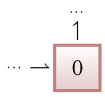
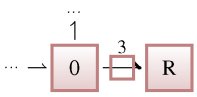
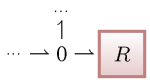
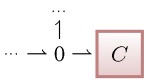
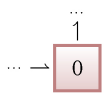
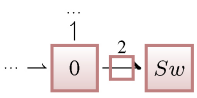
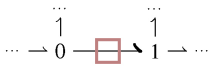
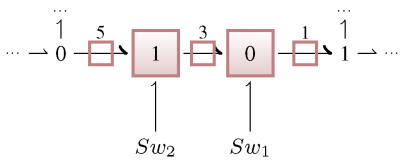
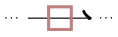
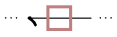
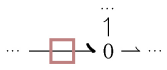
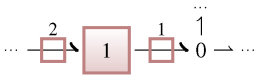
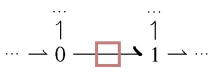
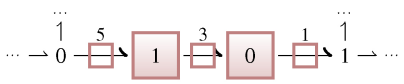
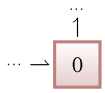
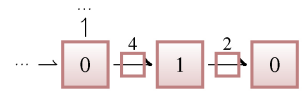
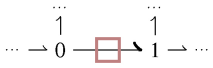
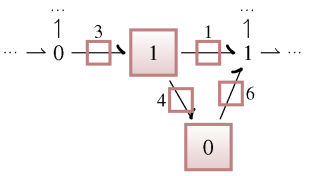
Also, the structure space formed by the GP tree is a highly non-linear space, as a change of a single node can have a drastic impact on the behavior of the phenotype due to the weak causality of GP [36]. As a result, distance metrics based on the GP tree are not very efficient at predicting similarities in performance.

With structure and parameter optimization in mind, techniques called SFS [29] and HFC [37], [38] were proposed. In traditional fitness sharing, for instance, the population is distributed over multiple peaks in the search space; in the neighborhood of each peak, a number of individuals proportional to the height of the related peak is allowed. In the same manner, SFS considers each GP tree structure as a peak in the space of structures. It tracks operations that alter the structure, as opposed to the parameter values, of the bond graph, and marks a structure as a new "peak" only when its bond graph structure changes. This allows one to control the amount of parameter optimization that occurs around each new structure and hence achieve a good balance between structure and parameter exploration.

HFC has proved to be an efficient approach to fight premature convergence. It allows a good balance between structure and parameter optimization by allowing structures with under-optimized parameters to compete in lower-fitness subpopulation classes. However, while delivering important performance improvements, HFC is not completely immune to structural convergence, especially if there are broad local minima in the search space that attract all most solutions. Therefore, it is the practice here to combine HFC with a version of structure fitness sharing that compares simplified bond graph structures.

The proposed SFS works in the same way as proposed in [29], but fully compares bond graphs obtained after executing the GP tree and the simplification algorithm. As a result, individuals are classified into species and their fitness

TABLE VI
BOND GRAPH MODIFICATION PRIMITIVES

Primitive	Before	After
Add Component		
Replace Component		
Add Switch		
Add Mutually Exclusive Switches		
Flip Bond		
Insert Junction		
Insert Junction Pair		
Add Junction Pair		
Split Bond		

can be adjusted according to the properties of the species to which they belong. As in SFS, the fitness adjustment factor of individual i is computed according to the size of its species, n_s^i, as follows:

$$\eta_{SFS}^i = \left(\frac{n_s^i}{N}\right)^{-\alpha} \tag{2}$$

where N is the number of allowed individuals per species and α is a parameter usually set to 1.5.

Inspired by the neuro-evolution approach NEAT [39], an aging factor is also added to the fitness computation. This gives the opportunity to remove strong individuals that have reached a dead end, like a strong local optimum. The aging adjustment factor of individual i is computed as follows:

$$\eta_{\text{age}}^i = \begin{cases} 1 - \dfrac{a_s^i - A}{\beta}, & \text{if } a_s^i > A \\ 1, & \text{otherwise} \end{cases} \tag{3}$$

where a_s^i is the age, in generations, of species s to which the individual i belongs, A is the threshold at which age is affecting the fitness, and β is a parameter defining the aging speed. If $\eta_{\text{age}}^i < 0$, then $\eta_{\text{age}}^i = 0$.

The final adjustment factor is obtained by multiplying both factors

$$\eta^i = \eta_{SFS}^i \eta_{\text{age}}^i \tag{4}$$

$$\text{If } \eta^i > 1, \text{ then } \eta^i = 1. \tag{5}$$

Finally, the fitness of the individual, Φ, is adjusted according to its adjustment factor

$$\Phi^i = \eta^i \Phi^i. \tag{6}$$

In summary, the individual fitness adjustment factor is only based on the size and age of its species. There is no measure of

TABLE VII
PARAMETERS FOR THE HAND-DESIGNED DC-DC CONVERTER

	Fitness	I	C_1	C_2
Initial	176.047	75 μH	800 μF	146.6 μF
Optimized	604.366	429 μH	401.128 μF	103.683 μF

distance between individuals. Only the corresponding species need to be established by comparing the simplified bond graphs.

IV. CASE STUDY

The single-input double-output DC-DC converter is used to demonstrate the capabilities of the proposed approach to evolve hybrid systems. An interesting aspect of this particular case study is that traditional circuit designs already exist with which to compare the performance of the evolved circuit.

A. Hand-Designed Topology

The traditional circuit against which the new design is compared was proposed in [40]. The converter circuit is reprinted here in Fig. 6(a) and its bond graph representation is shown in Fig. 6(b).

The initial parameters given in [40] have been optimized in order to ensure a fair comparison with this circuit topology. The parameter optimization process was done within the HBGGP framework by simply freezing the given topology and allowing only modifications of the parameter-related nodes in the GP tree. This can be done by initializing the population with the GP-tree shown in Fig. 6(c) and setting the parameter modification bias to 1. Consequently, only the three parameter nodes of the initial GP tree will be allowed to be replaced by numeric sub-trees.

Table VII lists the parameters obtained at the end of the optimization using the training case values listed in Table IX. The simulation period is set to 20 ms, to accommodate a typical settling time for the studied DC-DC converter, given the initial parameter taken from [40]. It is notable from Table IX that while the input voltage and the target voltage outputs remain the same during the simulation period, the two resistors change their values for three times. The training case is set up in this way so that we can also test the behavior of the evolved circuit design under changing load conditions. This is further explained in Section IV-C. Fig. 7 shows side-by-side the behavior of the system before and after the parameter optimization. The dotted red line in this figure and the remaining figures of this article represents the desired target values.

The circuit with new parameters effectively reduces the initial transient phase and the current ripple. Hence, the fitness was greatly enhanced with the new component values.

B. Design of the Embryo

To evolve bond graphs, an embryo must be defined. In our design, the embryo has a single voltage input, v_{in}, and two voltage outputs, v_1 and v_2. Two load resistances, R_1 and R_2,

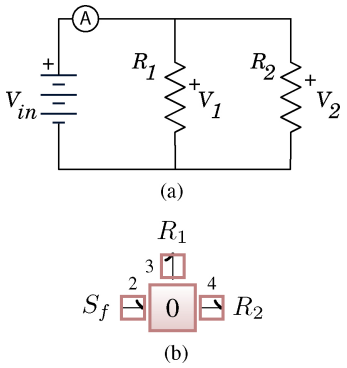


Fig. 8. Embryo of the DC-DC converter circuit. (a) Schematic of the embryo circuit. (b) Bond graph representation of the embryo circuit.

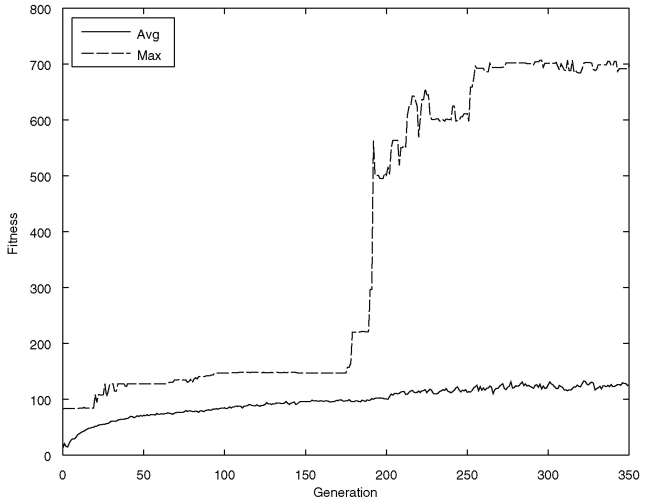
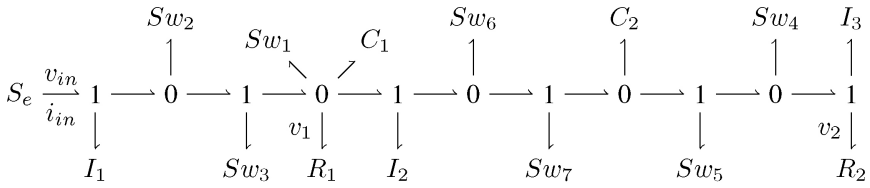


Fig. 9. Progression of the population fitness during the evolution of both the parameters and topology of the DC-DC converter.

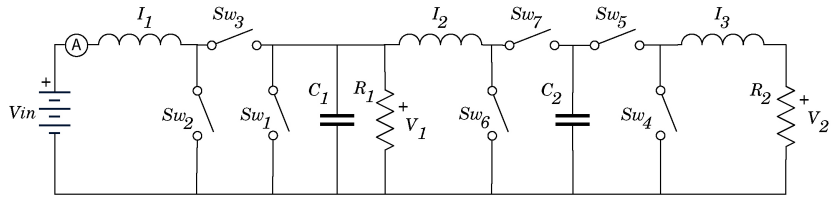
are used to model the loads to which the system is connected. This description of the system interface leads to the embryo represented in Fig. 8, in which Fig. 8(b) is a bond graph representation of the schematic of the embryo circuit described in Fig. 8(a). The embryo has four modification sites. The first one is located on the 0-junction and the three others are distributed on each bond.

C. Variable Load Resistances

In order to avoid the emergence of a solution that would be tuned only for a specific load case, the load resistance values are changed three times during the simulation. In fact, in early experiments when the change of load resistance was not introduced, the typical evolved designs did not involve any switching at all, and only used fine-tuned capacitors and inductors. This is because with carefully selected parameters, these components can still form a well-behaving circuit to achieve the desired outputs. However, as soon as the parameters of the components are slightly changed, the system performance breaks down. Therefore, changes in value of the load resistance have been applied in simulated training, to effectively press the evolution to propose solutions that are not heavily dependent on specific parameter values, forcing the evolved system to adopt a switching approach.



(a)



(b)

I_1	I_2	I_3	C_1	C_2
408.247 µH	544.945 µH	11.1625 mH	139.620 µF	69.5661 µF

(c)

Fig. 10. Best DC-DC converter at the end of the evolution. (a) Bond graph of the best individual at the end of the evolution. (b) Interpretation of the bond graph. (c) Parameter values.

TABLE VIII
PARAMETERS USED TO EVOLVE THE DC-DC CONVERTER SYSTEM

Objective	Find a single-input electrical system that minimizes the error on two outputs
Topology function set	ReplaceC, ReplaceI, ReplaceSwitch, RemoveComponent, AddSw, AddC, AddI, InsertJ01, InsertJ10
Parameter function set	MUL, DIV, ADD, SUB
Terminal set	EndBond, EndNode, EndJct, Ephemeral
Fitness	Maximum sum squared error to target of the two outputs
Selection	HFC with tournament size 2
Termination	Maximum number of generation reached
Parameters	Population size of 300 and 7 × 100, 0.85 parameter modification, 0.9 crossover, 0.10 standard mutation, 0.05 shrink tree mutation, 0.05 swap mutation, simulation time step of 0.01 ms. Maximum of 7 switches allowed

TABLE IX
TRAINING CASE VALUES

t	V_{in}	R_1	R_2	$\hat{V}_1$	$\hat{V}_2$	$\hat{I}$
0	1.5	6.25	34.1	1.875	3.75	0.650
5e-3	1.5	10	50	1.875	3.75	0.422
10e-3	1.5	14	40	1.875	3.75	0.402
15e-3	1.5	20	100	1.875	3.75	0.211

D. Experimental Setup

The evolution was conducted using the parameters of GP listed in Table VIII. The topology modification set included only functions to add switches and energy storage components, because an ideal power converter should not dissipate any energy. On the other hand, the model accuracy can be increased by using non-ideal energy storage components, such that parallel and serial resistances are added to each capacitor and inductor. In this case, the energy storage component and its accompanying resistances should therefore be seen as a compound component. However, the added complexity has not been judged necessary to demonstrate the evolvability of the circuit, and therefore it has not yet been implemented in this research.

A strong bias toward the parameter search is used to allow enough parameter tuning for every new topology, while HFC is used to keep structural diversity in the population.

The number of switches in the circuit is limited to a maximum of seven. Individuals with more than seven switches receive a fitness value of zero. This prevents the explosion of the simulation time required to evaluate the proposed solutions.

To evaluate the fitness of the evolved controllers, the system is integrated for a period of 20 ms. The outputs are then compared with the targets listed in Table IX. The same targets apply for all training cases in this paper.

The current target is calculated using the resistances and voltages according to the following equation:

$$I = \frac{V_1^2}{V_{in} R_1} + \frac{V_2^2}{V_{in} R_2}. \tag{7}$$

An objective function $\phi(k)$ is computed for each output at the end of the simulation, based on their errors

$$\phi(k) = \int (y_k - T_k)^2 dt : k \in 1, 2, 3 \tag{8}$$

where $y = \{V_1, V_2, I\}$ are the actual output values, and $T = \{\hat{V}_1, \hat{V}_2, \hat{I}\}$ are the target ones.

Later, the fitness for this controller is defined as the worst objective function

$$\Phi = \frac{1}{\max(\phi(k))} : k \in 1, 2, 3. \tag{9}$$

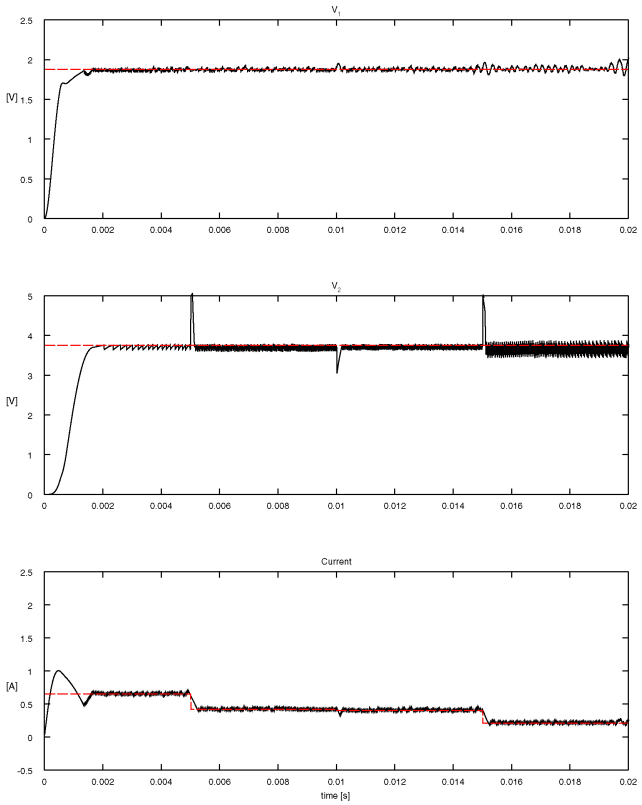


Fig. 11. Performance of the best evolved single-input double-output DC-DC converter of Fig. 10.

Looking at the worst case proved to be the best approach to push the search algorithm to reach the desired target. In fact, when looking at the average, the incentives to reach the targets were not strong enough, and the evolution tended to cease after finding a compromise between the errors of the two outputs.

E. Results

The time of a single run of the algorithm was in the range of a few days. The simulation time was greatly increased compared to the original BGGP approach which had a runtime of a few hours. The high frequency and switching behaviors presented in the case study require a much smaller simulation time step that drastically increases the computational cost of the fitness evaluation using the library of hybrid bond graph simulation we developed in the study.[1] The library can be improved in a future study by adopting a better step length control algorithm. The progression of the population fitness during the evolution of the DC-DC converter is plotted in Fig. 9. A significant breakthrough in the structure design occurred somewhere between the 189th and 192nd generation, at which time the fitness jumped over 500. Afterwards, the fitness progressed, mainly due to parameter optimization, until a final plateau was reached.

The best DC-DC converter design found using the evolutionary approach is the one shown in Fig. 10 and its behavior on the training case is plotted in Fig. 11.

[1]This library is available at http://sourceforge.net/projects/libbondgraph.

TABLE X
OPEN STATE RATIO OVER ALL ACHIEVED SWITCH CONFIGURATION FOR THE EVOLVED DC-DC CONVERTER

Sw_1	Sw_2	Sw_3	Sw_4	Sw_5	Sw_6	Sw_7
1.000	0.815	0.185	0.814	0.186	0.433	0.567

TABLE XI
FITNESS OBTAINED ON THE VALIDATION SET FOR THE EVOLVED DC-DC CONVERTER OF FIG. 10

Case	A	B	C	Gain
1	85.56	61.84	356.47	270.91
2	83.59	179.29	343.73	164.44
3	133.97	73.43	314.71	180.74
4	111.85	407.43	457.85	50.42
5	124.17	62.06	355.57	231.40
6	127.55	113.10	383.32	255.77
7	118.14	112.62	446.45	328.31
8	119.36	95.95	448.72	329.36
9	92.96	170.86	412.37	241.51
10	80.64	74.31	407.02	326.38
11	123.58	96.32	105.10	-18.48
12	96.24	56.25	336.40	240.16
13	80.89	150.75	321.14	170.39
14	126.97	51.93	324.22	197.25
15	116.46	54.93	115.01	-1.45
16	113.10	166.86	277.83	110.97
17	94.43	326.34	413.18	86.84
18	101.54	100.61	440.98	339.44
19	116.59	66.96	113.08	-3.51
20	90.56	73.64	116.34	25.78
21	155.51	250.68	295.79	45.11
22	104.88	235.99	427.16	191.17
23	107.75	56.26	217.27	109.52
24	91.54	113.85	379.79	265.94

A: hand-designed topology, hand-designed parameters.
B: hand-designed topology, evolved parameters.
C: evolved topology, evolved parameters.

It can be seen that the proposed design successfully generates the desired output with an acceptable ripple level. In addition, the initial rise time is reduced to a large extent. The discontinuity in the load resistance introduces high voltage peaks that are, however, quickly damped.

The design also exploits the maximum number of switches allowed to give a maximum of flexibility in the system control. However, from inspection of the circuit diagram of Fig. 10(b), Sw_1 and Sw_2 are redundant. One of these two could be removed without any effect. This is confirmed by the switch states analysis with results listed in Table X. Sw_1 is always kept open, i.e., no current passes through it.

The control flexibility offered by this design is highlighted by the impressive performance on the randomly generated cases that were not used during the evolution. In Table XI, the performance of the evolved design with both evolved topology and evolved parameters (case C) is compared against that of the hand-designed circuit using either the hand-designed parameters (case A) or the evolved parameters (case B). The gain in the table is the performance difference between the evolved circuit (case C) and the better of the two hand-designed ones (cases A and B). It can be easily seen that the hand-designed circuit with evolved parameters (case B) fails to obtain better performances than circuit case A in many of

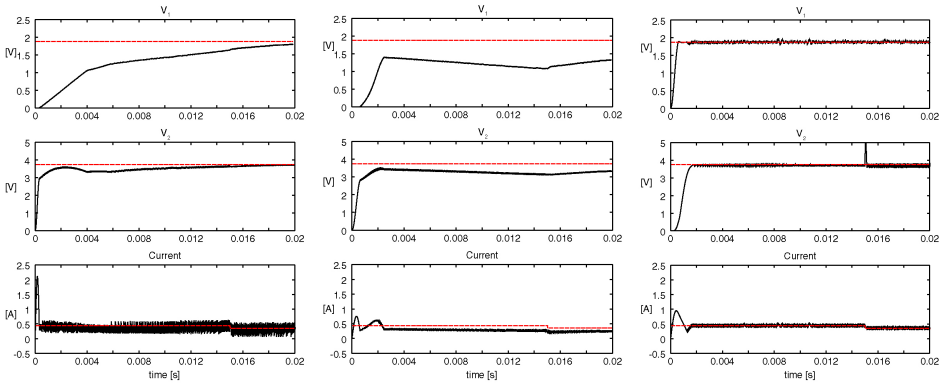


Fig. 12. DC-DC converter behaviors of case A, B, and C on case 18 of the validation set presented in Table XI.

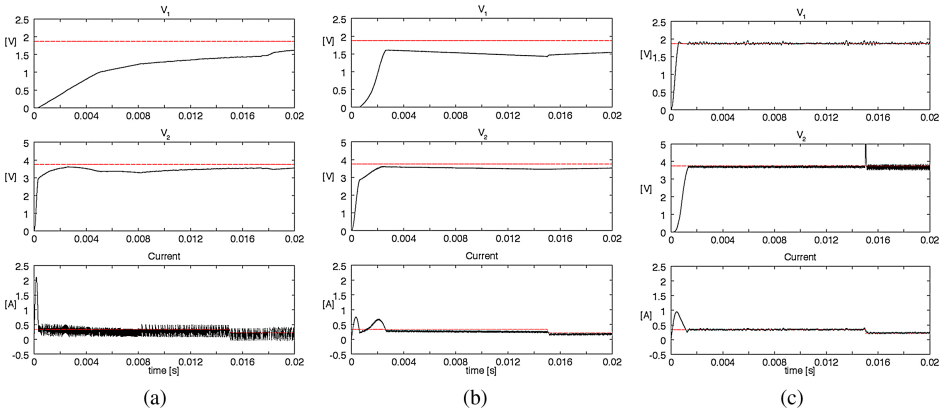


Fig. 13. DC-DC converter behaviors of case A, B, and C on case 13 of the validation set presented in Table XI.

the test cases, even though it performs much better in the training case. In contrast, the circuit case C shows a much more consistent performance improvement over circuit case A. A selection of output behaviors for these cases can be seen in Figs. 12–14, where each figure shows the output behaviors of three circuit cases A, B, and C according to a set of selected load conditions. To cover a complete spectrum, we included the best and the worst cases, as well as an average one. In each figure, the first subfigure A relates to the output behavior of the circuit case A, subfigure B to the circuit case B, and subfigure C to the circuit case C.

In terms of the fitness definition adopted in this paper, the evolved design outperforms by a significant margin both hand-designed ones in all except three cases. In those cases, the evolved design is outperformed by a slight margin by the original hand-design (case A), due to an increase in ripple for the last load resistance combination. Fig. 14 shows the worst performance on the validation set, where the last target values introduce some instabilities.

In defining the fitness function, this paper takes into account only the differences of the current and voltage outputs from their target values. Thus, it ignores many other possibly relevant considerations such as amount of ripple, suppression of spikes, and even circuit complexity. Although it will be important to include such considerations in a multiobjective approach, we chose to leave that as a topic for future research. In the first step taken here, the focus is on finding an approach that can generate novel designs automatically, based on a predefined design objective.

F. Generation of Alternative Design Concepts

Being a stochastic process, each independent evolutionary run can converge to a different solution. For example, the design shown in Fig. 15 was obtained from another evolution run. This design performs even better than the design of Fig. 10 in every single case on both the training and validation sets, as shown in Table XII.

It can also be seen from Fig. 16 that the initial rise time of both voltage outputs is reduced to less than 1 ms. This is a significant improvement because the traditional human design has a typical rise time of more than 10 ms. The ripples are further compressed in the voltage outputs, especially for V_2. These results show that with the method proposed in this paper, given a user-defined specification, we can automatically generate novel designs that have a striking different characteristic compared to the traditional design.

This also illustrates how the proposed approach can be applied multiple times to generate a set of solution concepts that the designer can eventually choose among.

G. Observations

This case study demonstrates that the HBGGP method can generate novel, competitive designs based on predefined

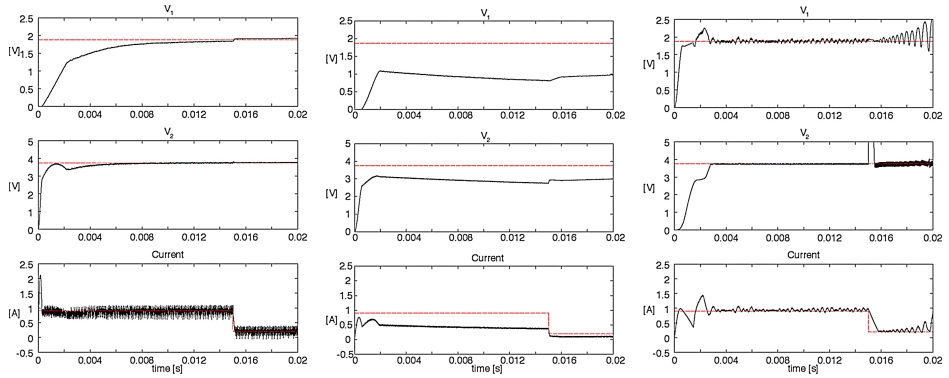
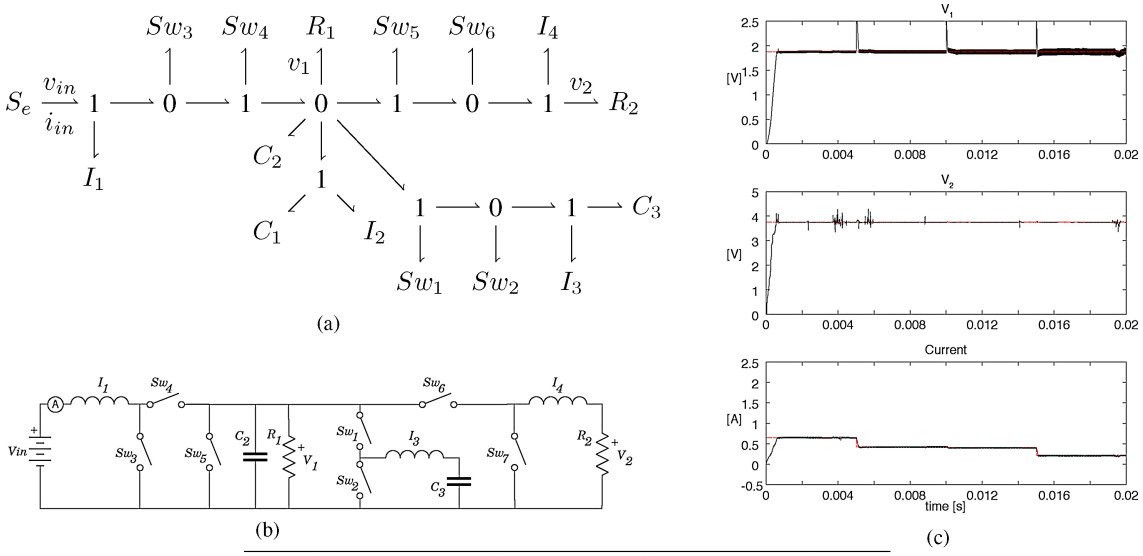


Fig. 14. DC-DC converter behaviors of case A, B, and C on case 19 of the validation set presented in Table XI. This is the worst behavior observed on the validation set.



I_1	I_2	I_3	I_4
1.136 47 mH	1.563 79 mH	6.2203 µH	3.010 84 mH

C_1	C_2	C_3
1.277 06 µF	563.893 µF	692.809 pF

(d)

Fig. 15. Alternate DC-DC converter design obtained from a second evolution run. (a) Bond graph of the best individual at the end of the evolution. (b) Interpretation of the bond graph. (c) Performance on the training case. (d) Parameter values.

design specifications. This is achieved by optimizing both the topology and the parameters. It is notable that even though optimizing only the parameters could generate better design for a specific load condition, it was not able to improve the performance in many other load conditions tested. This means that the parameter optimization is over-trained for a specific load condition in our case study. On the other hand, optimizing the parameters and meanwhile allowing the structure of the circuit to be changed has made it possible to design novel circuits that can deliver robustly good performance in a large variety of load conditions.

The fitness function has a major impact on the resulting evolved designs. Here, only the integrated error to the target was taken into account in the fitness function. Hence, the errors caused by the start rising transition and those caused by ripple and transient voltage spikes are not differentiated. As a result, the generated designs minimize the initial rise time successfully, but also induce a certain amount of ripple and transient voltage spikes that are not more severely penalized during the evaluation of the circuit performance under the chosen fitness function.

The method could be improved by using a multiobjective approach in the fitness evaluation to introduce more design considerations. For example, the fitness function could include objectives dealing with reducing ripple, transient voltage spikes, as well as circuit complexity. It is also worthwhile to point out that the main purpose of this paper is not to find a DC-DC converter design that can outperform the current human design in all aspects. Instead, we proposed a methodology that can generate novel designs based on

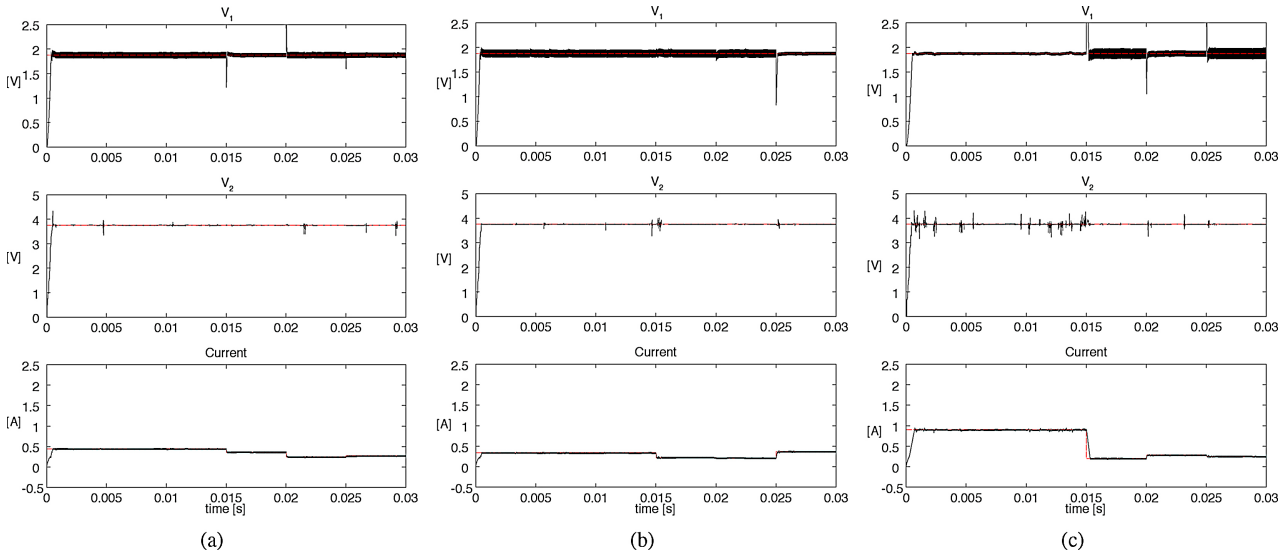


Fig. 16. Performance of the alternative DC-DC converter design of Fig. 15 on validation set cases 13, 18, and 19.

TABLE XII
COMPARISON OF THE FITNESS OBTAINED ON THE TRAINING AND VALIDATION SET FOR THE TWO EVOLVED DC-DC CONVERTERS

Case	Fitness (Fig. 10)	Fitness (Fig. 15)
Training	709.41	1089.71
1	356.47	661.82
2	343.73	716.46
3	314.71	872.68
4	457.85	903.87
5	355.57	748.64
6	383.32	903.02
7	446.45	715.97
8	448.72	766.36
9	412.37	842.03
10	407.02	692.42
11	105.10	598.84
12	336.40	658.80
13	321.14	740.89
14	324.22	747.94
15	115.01	405.98
16	277.83	773.11
17	413.18	865.58
18	440.98	881.33
19	113.08	580.18
20	116.34	582.97
21	295.79	777.65
22	427.16	826.28
23	217.27	586.98
24	379.79	777.50

predefined design specification, which paves the way for more detailed design considerations to be integrated in a practical real world design. This will be a major work of the next step in an attempt to extend the current method using multiobjective evolutionary computation.

V. CONCLUSION

The paper proposed an evolutionary method for automated design of hybrid mechatronic systems. The method, HBGGP, combines hybrid bond graphs representing both discrete and continuous dynamics for mechatronic systems, with genetic programming as a strong search tool to explore the open-ended design space, optimizing topologies and parameters simultaneously. Lookahead controllers are implemented as the default controller to control the discrete events of the system. One good feature of a lookahead controller is that its design procedure is very straightforward and can be conveniently automated in the loop of an evolutionary design algorithm. A case study of design of a DC-DC converter circuit shows that the HBGGP approach can evolve designs with novel topologies that are independent of human design experiences. Because the parameter sizing of the evolved design topologies is also automatically and properly determined by genetic programming, the evolved novel circuits can achieve improved performance in terms of criteria embedded in the fitness function and predefined by the user.

REFERENCES

[1] H. Coelingh, T. D. Vries, and J. V. Amerongen, "Automated conceptual design of mechatronic systems," *Journal A*, vol. 38, no. 3, pp. 26–29, 1997.

[2] J. Cagan, M. I. Campbell, S. Finger, and T. Tomiyama, "A framework for computational design synthesis: Model and applications," *J. Comput. Inform. Sci. Eng.*, vol. 5, pp. 171–181, Sep. 2005.

[3] M. I. Campbell, J. Cagan, and K. Kotovsky, "A-design: An agent-based approach to conceptual design in a dynamic environment," *Res. Eng. Des.*, vol. 11, pp. 172–192, Oct. 1999.

[4] T. Kurtoglu and M. I. Campbell, "Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping," *J. Eng. Des.*, vol. 20, no. 1, pp. 83–104, Feb. 2009.

[5] Z. Fan, "Design automation of mechatronic systems using evolutionary computation and bond graph," Ph.D. dissertation, Dept. Electr. Comput. Eng., Michigan State Univ., East Lansing, MI, 2004, adviser E. D. Goodman.

[6] J. Hu, Z. Fan, J. Wang, S. Li, K. Seo, X. Peng, J. Terpenny, R. Rosenberg, and E. Goodman, "GPBG: A framework for evolutionary design of multi-domain engineering systems using genetic programming and bond graphs," in *Design by Evolution*. Berlin/Heidelberg, Germany: Springer, 2008, pp. 319–345.

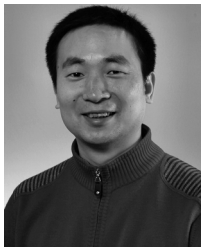
[7] K. Seo, Z. Fan, J. Hu, E. D. Goodman, and R. C. Rosenberg, "Toward a unified and automated design methodology for multi-domain dynamic systems using bond graphs and genetic programming," *Mechatronics*, vol. 13, nos. 8–9, pp. 851–885, 2003.

[8] Z. Fan, K. Seo, J. Hu, E. D. Goodman, and R. C. Rosenberg, "A novel evolutionary engineering design approach for mixed-domain systems," *Eng. Optimiz.*, vol. 36, no. 2, pp. 127–147, 2004.

[9] Z. Fan, J. Wang, and E. Goodman, "Exploring open-ended design space of mechatronic systems," *Int. J. Adv. Robot. Syst.*, vol. 1, no. 4, pp. 295–302, 2004.

[10] J. Hu, E. D. Goodman, S. Li, and R. Rosenberg, "Automated synthesis of mechanical vibration absorbers using genetic programming," *Artif. Intell. Eng. Des., Anal. Manuf.*, vol. 22, no. 3, pp. 207–217, 2008.
[11] J. Wang, Z. Fan, J. P. Terpenny, and E. D. Goodman, "Knowledge interaction with genetic programming in mechatronic systems design using bond graphs," *IEEE Trans. Syst., Man Cybern., Part C*, vol. 35, no. 2, pp. 172–182, May 2005.
[12] J. Wang, Z. Fan, J. P. Terpenny, and E. D. Goodman, "Cooperative body-brain coevolutionary synthesis of mechatronic systems," *Artif. Intell. Eng. Des., Anal. Manuf.*, vol. 22, no. 3, pp. 219–234, 2008.
[13] Z. Fan, J. Wang, S. Achiche, E. Goodman, and R. Rosenberg, "Structured synthesis of MEMS using evolutionary approaches," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 579–589, 2008.
[14] J.-F. Dupuis, Z. Fan, and E. Goodman, "Evolved finite state controller for hybrid system," in *Proc. 1st ACM/SIGEVO Summit GEC*, 2009, pp. 105–112.
[15] J.-F. Dupuis and Z. Fan, "Evolved finite state controller for hybrid system in reduced search space," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, Jul. 2009, pp. 833–838.
[16] J.-F. Dupuis and Z. Fan, "Comparing an evolved finite state controller for hybrid system to a lookahead design," in *Proc. IEEE World CEC*, Jul. 2010, pp. 1–6.
[17] R. C. Rosenberg, "State-space formulation for bond graph models of multiport systems," *Trans. ASME J. Dyn. Syst.*, vol. 93, no. 1, pp. 35–40, 1971.
[18] D. Karnopp, D. Margolis, and R. Rosenberg, *System Dynamics: Modeling and Simulation of Mechatronic Systems*, 4th ed. Hoboken, NJ: Wiley, 2005.
[19] P. J. Mosterman, "Hybrid dynamic systems: A hybrid bond graph modeling paradigm and its application in diagnosis," Ph.D. dissertation, Vanderbilt Univ., Nashville, TN, 1997 [Online]. Available: http://macs.isis.vanderbilt.edu/publications/db/mosterman97_thesis.pdf
[20] U. Söderman and J.-E. Strömberg, "Switched bond graphs: Multiport switches, mathematical characterization and systematic composition of computational models," Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 1995 [Online]. Available: citeseer.ist.psu.edu/96874.html
[21] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.
[22] F. Gruau, "Genetic micro programming of neural networks," in *Advances in Genetic Programming*, vol. 1. Cambridge, MA: MIT Press, 1994, pp. 495–518.
[23] J. Lohn and S. Colombano, "A circuit representation technique for automated circuit design," *IEEE Trans. Evol. Computat.*, vol. 3, no. 3, pp. 205–219, Sep. 1999.
[24] S. Luke and L. Spector, "Evolving graphs and networks with edge encoding: Preliminary report," in *Proc. Late Breaking Papers Genet. Program. Conf.*, 1996, pp. 117–124.
[25] G. S. Hornby and J. B. Pollack, "Creating high-level components with a generative representation for body-brain evolution," *Artif. Life*, vol. 8, no. 3, pp. 223–246, 2002.
[26] A. Lindenmayer, "Mathematical models for cellular interactions in development, II, simple branching filaments with two-sided inputs," *J. Theor. Biol.*, vol. 18, no. 3, pp. 300–315, Mar. 1968.
[27] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction, On the Automatic Evolution of Computer Programs and Its Applications* (The Morgan Kaufmann Series in Artificial Intelligence). San Mateo, CA: Morgan Kaufmann, Nov. 1997.
[28] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving*, 1st ed. San Mateo, CA: Morgan Kaufmann, May 1999.
[29] J. Hu, K. Seo, S. Li, Z. Fan, R. C. Rosenberg, and E. D. Goodman, "Structure fitness sharing (SFS) for evolutionary design by genetic programming," in *Proc. GECCO*, 2002, pp. 780–787.
[30] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Dept. Comput. Sci., Univ. Michigan, Ann Arbor, 1975.
[31] R. Manner, S. Mahfoud, and S. W. Mahfoud, "Crowding and preselection revisited," in *Parallel Problem Solving from Nature*. Amsterdam, The Netherlands: North Holland, 1992, pp. 27–36.
[32] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Reading, MA: Addison-Wesley, Jan. 1989.
[33] E. D. de Jong and J. B. Pollack, "Multiobjective methods for tree size control," *Genet. Program. Evol. Mach.*, vol. 4, no. 3, pp. 211–233, Sep. 2003.
[34] A. Ekárt and S. Z. Németh, "A metric for genetic programs and fitness sharing," in *Proc. Eur. Conf. Genet. Program.*, 2000, pp. 259–270.
[35] R. I. McKay, "Fitness sharing in genetic programming," in *Proc. Genet. Evol. Computat. Conf.*, Jul. 2000, pp. 435–442.
[36] J. P. Rosca and D. H. Ballard, "Causality in genetic programming," in *Proc. 6th Int. Conf. Genet. Algorithms*, 1995, pp. 256–263.
[37] J. J. Hu and E. D. Goodman, "The hierarchical fair competition (HFC) model for parallel evolutionary algorithms," in *Proc. CEC*, vol. 1. 2002, pp. 49–54.
[38] J. Hu, E. Goodman, K. Seo, Z. Fan, and R. Rosenberg, "The hierarchical fair competition (HFC) framework for sustainable evolutionary algorithms," *Evol. Comput.*, vol. 13, no. 2, pp. 241–277, 2005.
[39] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.
[40] M. Senesky, G. Eirea, and T. J. Koo, "Hybrid modeling and control of power electronics," in *Proc. 6th Int. Workshop HSCC*, LNCS 2623. 2003, pp. 450–465.

Jean-François Dupuis received the B.Eng. degree in mechanical engineering and the M.S. degree in electrical engineering from Laval University, QC, Canada, in 2003 and 2007, respectively, and the Ph.D. degree in mechanical engineering from the Technical University of Denmark, Lyngby, Denmark, in 2011.



Zhun Fan (SM'10) received the B.Sc. and M.Sc. degrees in control engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1995 and 2000, respectively, and the Ph.D. degree in electrical engineering from Michigan State University, East Lansing, in 2004.

From 2004 to 2007, he was an Assistant Professor with the Technical University of Denmark, Lyngby, Denmark. From 2007 to 2011, he was an Associate Professor with the Technical University of Denmark. He is currently a Full Professor with the College of Electronics and Information Engineering, Tongji University, Shanghai, China. He has been a principle investigator of various projects sponsored by the Danish Research Agency of Science Technology and Innovation. His research is also supported by the National Science Foundation. His major research interests include evolutionary computation, intelligent control and robotic systems, robot vision and cognition, MEMS, design automation and optimization, intelligent power systems and transportation systems, and so on.

Dr. Fan is a member of ACM and ASME.

Erik D. Goodman received the Ph.D. degree in computer and communication sciences from the University of Michigan, Ann Arbor, in 1971.

He is currently the PI and Director of the BEACON Center for the Study of Evolution in Action, an NSF Science and Technology Center headquartered at Michigan State University (MSU), East Lansing, and funded beginning in 2010. He was an Assistant Professor of electrical engineering and systems science in 1972, an Associate Professor in 1978, and a Professor in 1984, all with MSU, where he also holds appointments in mechanical engineering and in computer science and engineering. He directed the Case Center for Computer-Aided Engineering and Manufacturing from 1983 to 2002, and MSU's Manufacturing Research Consortium from 1993 to 2003. He has co-directed MSU's Genetic Algorithms Research and Applications Group since its founding in 1993. He is the Co-Founder and Vice President of Red Cedar Technology, Inc., East Lansing, a firm that develops design optimization software for use in industry. His current research interests include application of evolutionary principles to solution of engineering design problems.

Dr. Goodman was the Chair of the Executive Board and a Senior Fellow of the International Society for Genetic and Evolutionary Computation from 2003 to 2005. He was the Founding Chair of ACM's Special Interest Group on Genetic and Evolutionary Computation, serving from 2005 to 2007. He was chosen the Michigan Distinguished Professor of the Year in 2009, by the Presidents Council, State Universities of Michigan.