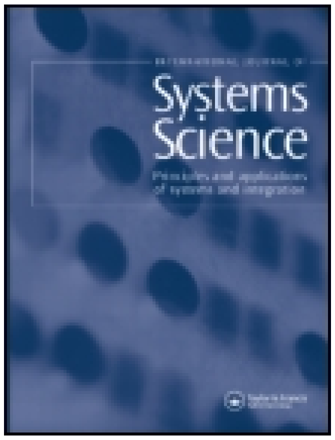


International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tsys20

Evolutionary design of discrete controllers for hybrid mechatronic systems

Jean-Francois Dupuis[a], Zhun Fan[b] & Erik Goodman[c]
[a] Department of Management Engineering at the Technical, University of Denmark, Kgs. Lyngby, Denmark
[b] Department of Electrical Engineering, Shantou University, Shantou, China
[c] BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI, USA

Published online: 28 May 2013.

To cite this article: Jean-Francois Dupuis, Zhun Fan & Erik Goodman (2015) Evolutionary design of discrete controllers for hybrid mechatronic systems, International Journal of Systems Science, 46:2, 303-316, DOI: 10.1080/00207721.2013.783643

To link to this article: http://dx.doi.org/10.1080/00207721.2013.783643

International Journal of Systems Science, 2015
Vol. 46, No. 2, 303–316, http://dx.doi.org/10.1080/00207721.2013.783643

Evolutionary design of discrete controllers for hybrid mechatronic systems

Jean-Francois Dupuis[a], Zhun Fan[b,*] and Erik Goodman[c]

[a]*Department of Management Engineering at the Technical University of Denmark, Kgs. Lyngby, Denmark;* [b]*Department of Electrical Engineering, Shantou University, Shantou, China;* [c]*BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI, USA*

(*Received 3 November 2011; final version received 8 January 2013*)

This paper investigates the issue of evolutionary design of controllers for hybrid mechatronic systems. Finite State Automaton (FSA) is selected as the representation for a discrete controller due to its interpretability, fast execution speed and natural extension to a statechart, which is very popular in industrial applications. A case study of a two-tank system is used to demonstrate that the proposed evolutionary approach can lead to a successful design of an FSA controller for the hybrid mechatronic system, represented by a hybrid bond graph. Generalisation of the evolved FSA controller to unknown control targets is also tested. Further, a comparison with another type of controller, a lookahead controller, is conducted, with advantages and disadvantages of each discussed. The comparison sheds light on which type of controller representation is a better choice to use in various stages of the evolutionary design of controllers for hybrid mechatronic systems. Finally, some important future research directions are pointed out, leading to the major work of the succeeding part of the research.

Keywords: evolutionary computation; hybrid systems

1. Introduction

Mechatronic systems are the complete integration of mechanics, electronics and information processing. Tight integration of these domains makes them highly dependent on each other. Design choices in one domain affect the performance of the other domains. Therefore, design of such a system usually requires iterations in each domain (Li, Zhang, and Chen 2001) in order to find an optimal balance between the basic mechanical structure, sensor and actuator implementations, digital information processing and overall control. In an effort to automate the generation of mechatronic systems spanning multiple domains, an approach that combines the bond graph (Karnopp, Margolis, and Rosenberg 2005) representation and genetic programming for search was proposed (Seo, Fan, Hu, Goodman, and Rosenberg 2003; Fan, Seo, Hu, Goodman, and Rosenberg 2004a; Fan, Wang, and Goodman 2004b; Wang, Fan, Terpenny, and Goodman 2005, 2008; Fan, Wang, Achiche, Goodman, and Rosenberg 2008). In this previous work, good results on a variety of case studies have been presented, including electrical filter design (Seo et al. 2003; Fan et al. 2004a), typewriter redesign (Fan et al. 2004b), co-design of controller and plant embodiment of a vehicle suspension system (Wang et al. 2005, 2008) and microelectromechanical systems (Fan et al. 2008). However, all the case studies presented were limited to time-driven systems. Since most mechatronic systems involve both time-driven and event-driven dynamics, study of evolutionary design of discrete controllers provides an important step toward automated design of hybrid mechatronic systems.

Such hybrid systems may be viewed as an extension of a classical time-driven system, typically modelled through differential or difference equations, with occasional discrete events causing a change in its dynamic behaviour. When such an event takes place, the system is thought of as switching from one operating mode to another. Hybrid or switched bond graph representations (Strömberg, Nadjm-Tehrani, and Top 1996; Mosterman 1997) have been proposed to model physical dynamic systems with this type of discontinuity. This hybrid system representation extends the normal bond graph by adding a switch element that acts either as a flow source or as an effort source, according to the current system state or controller action. With this type of modelling, a supervisory control system (Ramadge and Wonham 1987; Koutsoukos, Antsaklis, Stiver, and Lemmon 2000) can then provide a decision-making ability to control a plant in a compact way. The resulting hybrid system can be represented as in Figure 1 (Koutsoukos et al. 2000), where the hybrid bond graph represents the plant and the controller governs the state of the switch elements in the bond graph. The interface acts as a translator between the continuous space and the discrete controller space. The switches in the hybrid bond graph act as the actuators issuing the commands $u(t)$ to the plant based on the output of the

*Corresponding author. Email: zfan@stu.edu.cn

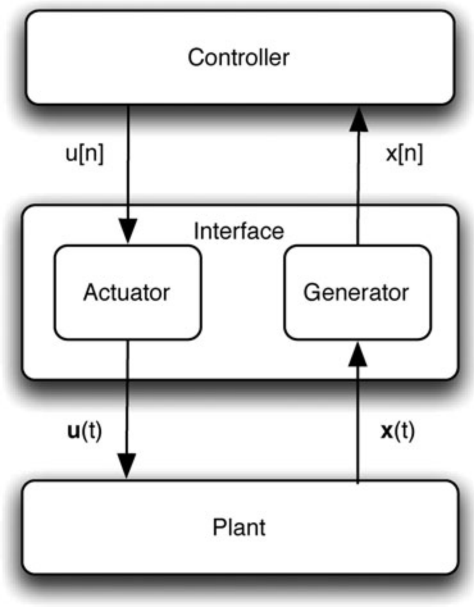


Figure 1. Representation of a general hybrid control system. In this work, the plant is modelled using a hybrid bond graph and the controller is represented by a finite state automaton.

discrete controller $u[n]$. The generator generates relevant discrete symbols, $x[n]$, to be interpreted by the controller from the continuous dynamic behaviour of the plant, which is represented by state $x(t)$.

This article presents the first part of the research that aims to extend the previous evolutionary design approach to hybrid systems that can be represented by hybrid bond graphs. In this part, we focus on investigating evolutionary design of controllers with a given plant. In the second part, we will extend the work to cases where the plant is open-ended and topology becomes evolvable. In both cases, controlling the switch reconfigurations in the hybrid bond graphs plays a critical role in achieving the desired overall performance of the system. It is important to note that the discrete controller for switch reconfiguration cannot be represented by a bond graph, and hence cannot be evolved as part of the overall bond graph structure covering both controller and plant, as done for the design of the active car suspension control system in Wang et al. (2008). It is therefore important to find a suitable evolvable representation for controllers of hybrid mechatronic systems.

Several techniques have been proposed over the years to automate discrete controller design, which is different by its nature from the discrete optimisation problem (Chen et al. 2010). For example, neural networks with evolvable topologies (Stanley and Miikkulainen 2002) can be used for controlling both time-driven and discrete-event systems. However, it is normally difficult to decipher or interpret a rule set from the neural network representation. Actually, for discrete controllers, finite state automata (FSA) are a more popular choice, which can also be used to model an adaptive fuzzy controller (Kung, Huang, and Tsai 2009), and are easily interpreted as a set of rules. The interest in evolution of FSA has a long history, starting almost 50 years ago with the work of Larry Fogel (Fogel 1962), and is still active today (Dunay, Petry, and Buckles 1994; Benson 2000; Ashlock 2006). In our work, we selected the FSA representation for the discrete controller due to its interpretability, fast execution speed and natural extension to a statechart, which is a very popular tool to describe discrete controllers in industrial applications.

Other approaches have also been proposed to deal with the specific aspects of hybrid system control (Stiver, Koutsoukos, and Antsaklis 2000; Bemporad 2003; Sun and Ge 2005a,b; Cormerais, Buisson, Richard, and Morvan 2008). For example, Sun and Ge (2005a) establish different conditions for stability and controllability of switched systems, and argue that the optimal control problem of switched systems is in general difficult to solve due to the involvement of the switching signals. Bemporad (2003) shows how model predictive control (MPC) could be applied to achieve optimal control solutions in the specific case of systems with multiple actuators and sensors, but with a fixed state matrix. Unfortunately, the systems represented by our hybrid bond graph cannot fall into this class because the reconfiguration of switches in our systems can induce changes in the state matrix.

Passivity-based control (PBC) (Ortega, Perez, Nicklasson, Sira-Ramirez, and Sira-Ramirez 1998) has also been successfully applied in many switching systems, especially in the field of power electronics. Cormerais et al. (2008) show how this can be applied from a hybrid bond graph representation of a multi-cellular converter, where switches commutate by pairs. In that case, all the configurations of the system can be represented by a single group of state equations (Buisson, Cormerais, and Richard 2002), allowing the PBC to be applied directly.

Another interesting approach to the control of a hybrid system is shown by Stiver, Koutsoukos, and Antsaklis (2001). This approach proposes to compute the natural invariants of the dynamical system from the systems characteristic equation, which are then used to partition the continuous state space. Once the divisions of the state space are defined, an FSA controller can be deduced. However, the solving of the characteristic equation is often computationally very expensive, if not impossible. Furthermore, the computed divisions are only valid for a specific target point. When the target is changed, a new partitioning needs to be made, which makes the approach extremely expensive to apply in practice.

In addition, all these approaches share a common feature of requiring human designers to conduct an intensive analysis of the control system based on very precise modelling, which makes them less interesting to us because this presents great difficulties for automated design. We, therefore, selected the lookahead controller, a member of the MPC family, for a comparison study with the FSA controller. In particular, we used a one-step lookahead controller to reduce the computational complexity.

In summary, the paper presents a comparative study of two controller representations in the framework of evolutionary design of both controller and plant of a hybrid dynamic system, in order to explore which representation better promotes evolvability of a useful design, depending on the formulation of the problem. The remainder of the paper is organised as follows. Section 2 describes the bond graph modelling of the two-tank system that was used as the case study. Then, Section 3 presents the work on evolutionary design of FSA controllers, while Section 4 presents the results obtained with the lookahead controller. Results of both approaches are compared and discussed in detail in Section 5. Finally, Section 6 concludes the paper and provides some important perspectives for future work.

2. Presentation of the case study of the two-tank system

Multiple-tank systems are often encountered in the research literature concerning non-linear multi-variable feedback control, as well as in fault diagnosis (Sainz, Armengol, and Vehi 2002; Wu, Biswas, Abdelwahed, and Manders 2005). The mechanical simplicity and the ease of getting physical insight into the system behaviour, combined with the achievable control complexity, make the multi-tank problem a very attractive testbed. A two-tank system was defined to test the controller synthesis methodology presented in this paper. Figure 2 shows the actual configuration. A pump is continuously filling the left tank at a constant flow rate, and a set of valves allows draining of each tank independently and also allows bidirectional transfer from one tank to the other.

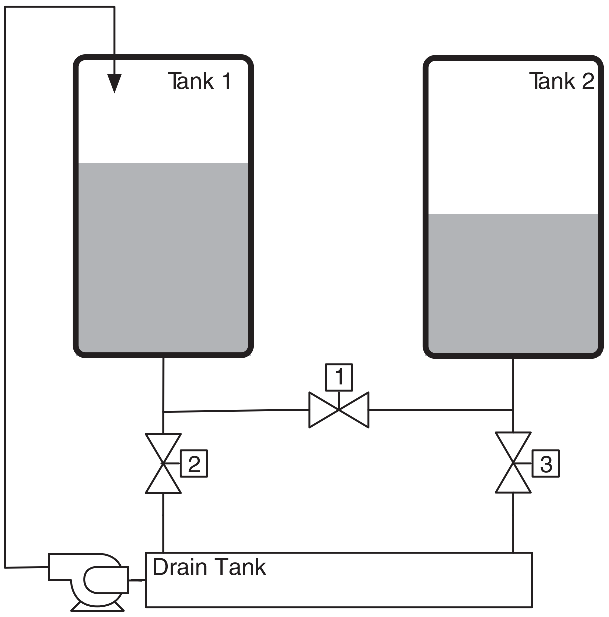


Figure 2. The two-tank system.

$$\begin{tikzcd}
 & C_1 & R_1 & C_2 & \\
Sf \arrow[r, "f"'] & 0 \arrow[u] \arrow[r] \arrow[d] & 1 \arrow[u] \arrow[r] \arrow[d] & 0 \arrow[u] \arrow[d] & \\
Sw_2 & 1 \arrow[l] \arrow[d] & Sw_1 & 1 \arrow[r] \arrow[d] & Sw_3 \\
 & R_2 & & R_3 &
\end{tikzcd}$$

Figure 3. The hybrid bond graph of the two-tank system.

The hybrid bond graph of this two-tank system is shown in Figure 3. The pipe and valve restrictions are represented by resistive components, *R*, while the tanks are represented by the capacitances, *C*. The input from the pump is modelled as a flow source S_f and the valves are modelled by switch components, *Sw*. These switches act as a 0-flow source or a 0-effort source, depending on their states. A 0-flow source imposes a flow equal to zero at the junction connected to it, therefore the valve is said to be closed as no fluid is able to pass through it. A 0-effort source indicates that the switch does not impose any restriction on the flow. The valve is then said to be open.

The vector state equation of the two-tank system can be obtained from the hybrid bond graph :

$$\begin{aligned}
\dot{x_1}(t) &= f - \frac{u_1(t)}{R_1}\left(\frac{x_1(t)}{C_1} - \frac{x_2(t)}{C_2}\right) - \frac{u_2(t)}{R_2}\frac{x_1(t)}{C_1} \\
&= g_1(\mathbf{x}(t), \mathbf{u}(t)) \\
\dot{x_2}(t) &= \frac{u_1}{R_1}\left(\frac{x_1(t)}{C_1} - \frac{x_2(t)}{C_2}\right) - \frac{u_3(t)}{R_3}\frac{x_2(t)}{C_2} \\
&= g_2(\mathbf{x}(t), \mathbf{u}(t)),
\end{aligned} \tag{1}$$

where *x* denotes the state vector and *u* the input vector. The level *y* of tank *i* can be obtained from the state variables :

$$y_i = \frac{x_i(t)}{C_i \rho g}, \tag{2}$$

where ρ is the fluid density and *g* is gravity. This system of equations can also be expressed using the matrix formulation:

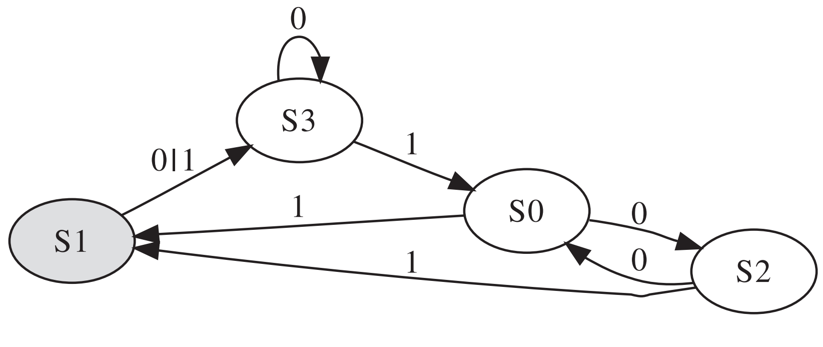
$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \begin{bmatrix} -\frac{u_1(t)}{R_1C_1} - \frac{u_2(t)}{R_2C_1} & \frac{u_1(t)}{R_1C_2} \\ \frac{u_1(t)}{R_1C_1} & -\frac{u_1(t)}{R_1C_2} - \frac{u_3(t)}{R_3C_2} \end{bmatrix} \mathbf{x}(t) + \begin{matrix} f \\ 0 \end{matrix} \\
\mathbf{y}(t) &= \operatorname{diag} \frac{1}{C_1\rho g} \; \frac{1}{C_2\rho g} \, \mathbf{x}(t).
\end{aligned} \tag{3}$$

	0	1
0	2	1
1	3	3
2	0	1
3	3	0
Initial = 1		

(a) Transition table.

10 01 11 11 00 01 11 00 01

(b) Genotype



(c) Phenotype

Figure 4. Encoding example. (a) Transition table, (b) genotype and (c) phenotype.

Therefore, the equation in state-space form for this two-tank system is:

$$\dot{\mathbf{x}}(t) = A(\mathbf{u}(t))\mathbf{x}(t) + B$$
$$\mathbf{y}(t) = C\mathbf{x}(t). \tag{4}$$

3. Evolved FSA controller

3.1. Encoding scheme

Evolutionary computation has been applied in controller design in many applications (Wai and Tu 2007; Yu and Kaynak 2009; Samosir and Yatim 2010). In this work, a genetic algorithm is used to evolve the controller, and the encoding scheme used in this paper is adopted from Ashlock (2006). The controller trained in our research is an FSA having a fixed number of states, *N*. Each state corresponds to a possible switch configuration in the hybrid bond graph. In fact, each bit of the binary representation of a state's numeric value is associated with an individual switch.

The transitions used in this implementation of the FSA do not have any actions associated with them; they simply specify what the new state will be, in reaction to the input symbol received. The state of the FSA directly defines the 'ON' / 'OFF' status of an individual switch, and thus the behaviour of its linked actuator.

As a result, the controller can be expressed as a matrix with current states as row indices and received input symbols as column indices. The elements of the matrix then specify the next state to which the FSA should move. An initial state must also be defined outside this matrix to complete the definition.

The transition table matrix is encoded in a bit string with ceil($\log_2 N$) bits per element, and an extra set of bits at the end to define the initial state. With *N* states and *M* input symbols, the length of the evolved bit string is given by:

$$\text{ceil}(\log_2 N)(NM + 1). \tag{5}$$

Figure 4 shows the proposed scheme used in a simpler case with only four states and two input symbols. In this example, only two bits are needed for each state. One can see that the elements of the transition matrix are written row by row to the associated genotype, and the last two bits of the genotype indicate the initial state.

The controller associated with this example would be able to control a system having two switches. Looking at Figure 4 (c), we understand that in state 1 (01), the first switch would be turned off, while the other would be turned on. From this state, receiving either of the two input symbols would make the system migrate to state 3 (11), where both switches would be turned on. Then, receiving input symbol 1 would turn both switches off by establishing state 0 (00).

To evolve the controller, a simple genetic algorithm (GA) is used, represented by Figure 5, with standard one-point crossover, bit-flipping mutation and tournament selection. Elitism was not used in this case because run performed using elitism showed premature convergence that greatly reduced population diversity. The parameters of the

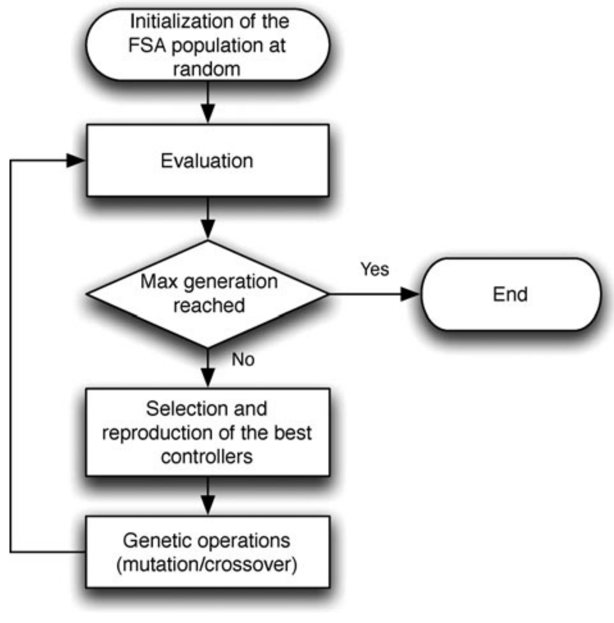

Initialization of the FSA population at random
Evaluation
Max generation reached
Yes
End
No
Selection and reproduction of the best controllers
Genetic operations (mutation/crossover)

Figure 5. Genetic algorithm flow chart.

evolutionary algorithm are summarised in Table 1. The GA implementation was done using the Open BEAGLE C++ framework (Gagné and Parizeau 2006) and the fitness evaluations were distributed on a cluster of computers using MPI (MPI 2008).

3.2. *FSA for the two-tank system*

For the two-tank system, the FSA controller has eight states with three switches. The input symbols are generated by the interface when the state variables cross predefined surfaces in the state space. These surfaces are separated into two sets, one for each tank, and are defined as follow:

$$h_{i1} = y_i - T_1 + \delta$$
$$h_{i2} = y_i - T_1 - \delta, \tag{6}$$

in which T_1 is the desired level for tank i and δ is the tolerance about the target. Each set separates the space into three regions depending on whether the level of tank i is above, below or at its target. Then nine symbols are formed from the logical conjunction of the two sets. With 9 input symbols, 8 states and 3 bits to represent each state, the bit string to be evolved is thus $(8 \times 9 + 1) \times 3 = 219$ bits long; note that we must also define an initial state in the string.

Table 1. Parameters used to evolve the FSA controller.

Objective:	Find a finite state controller that minimises the tank level errors.
Fitness:	Sum of squared error from target of the worst output, averaged over six training cases.
Selection:	Tournament with size 2.
Termination:	Maximum number of generations reached.
Parameters:	Population size 300, 0.3 one-point crossover, 0.01 bit-flip mutation.

3.2.1. *Fitness evaluation*

In the case of the two-tank system, the FSA controller was trained to keep the fluid levels of the tanks following their target profiles. When the fitness of the evolved controllers is evaluated, for each simulation case, the system is integrated for a period of 15 seconds. An objective function $\phi(i)$ is computed for each tank at the end of the simulation, based on the square of the level errors:

$$\phi(i) = \int_0^{15} (y_i - T_i)^2 dt, i \in 1, 2. \tag{7}$$

The fitness of this simulation case is then defined based on the larger objective function – i.e., the larger error of the two tanks:

$$\Phi = \frac{1}{\max\{\phi(i) \mid i \in (1, 2)\}}. \tag{8}$$

In our experiences, defining fitness in a maximisation/minimisation approach proved to be a more successful than using average error of the two tanks as fitness. When the average error was used for fitness, it was observed that most of the time, one of the tanks sacrificed its own performance for the other, which means that the evolved controller would rather focus on improving performance of only one of the tanks, while performance of the other was not improved at all. This is because the potentially poorer performance of the sacrificed tank could be compensated for by the performance of the other tank, and together they could still yield an improved average. On the other hand, if a fitness definition based on the worse performance is used, the evolution improves the performance of both tanks, disallowing the unwanted behaviour just described.

3.2.2. *Training and testing*

For evolutionary design of controllers, it is very important to divide the experimental process into a training phase and testing phase. The training phase evolves a controller according to some training targets, and the testing phase tests how well this evolved controller can generalise to other targets not involved in the training phase. In order for the evolved FSA controller to be able to follow a larger spectrum of different targets, usually more than one target should be used for training. In our experiments, we chose to look at the average performance obtained on the six randomly generated targets listed in Table 2 for training purposes. Each experiment used a single HP ProLiant

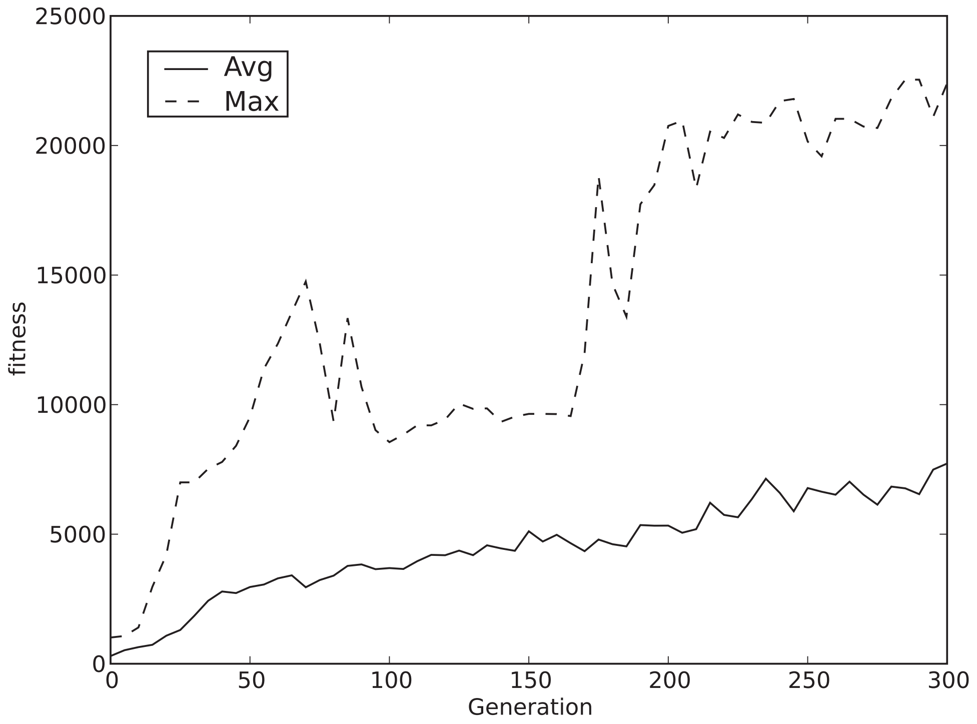


Figure 6. Progression of the population fitness during the evolution of the FSA controller for the two-tank system.

SL165z G7 with AMD Opteron Processor 6168 (twelve-core, 1.9 GHz, 12 MB L3 cache) with 64GB memory. It is worthwhile to point out that six may not be the optimal number of targets to use for training. But because our current work is only focused on demonstrating the feasibility of the evolutionary approach for controller design, we leave the further optimisation of the controller design to our future work.

During the training phase, the progression of the evolutionary process is shown in the fitness curve in Figure 6. It is interesting to look at the performance of the best controller obtained at the mid-point and the end of the evolution. The behaviour of the controller at the 150th generation is shown in Figure 8. Behaviour at the end of the evolution is shown in Figure 9. Comparison reveals that ripples existing in the middle stage of the evolution were largely eliminated at the end of the evolution. This comparison demonstrates that the proposed evolutionary design approach can evolve an FSA controller with incrementally improving performance to meet the predefined design targets. The best FSA controller obtained at the end of the evolution is shown in Figure 7 and its fitness for each training cases is listed in Table 3.

In the testing phase, the performances of the best evolved FSA controller, for which the phenotype is shown in Figure 7, was then tested on a set of 20 targets generated at random in order to verify whether the evolved controller was able to generalise to other control targets not involved in its training. Figure 10 shows the behaviour of the controller on the 20 targets of the verification set. As can be seen, the evolved controller generalises very well to the randomly generated targets, with only one failure, the 13th case, out of the 20 test cases.

Table 2. Training target used to evolved the FSA.

	$t = 0$		$t = 5$		$t = 10$	
Case	Tank 1	Tank 2	Tank 1	Tank 2	Tank 1	Tank 2
a	0.2	0.4	0.25	0.35	0.35	0.2
b	0.48	0.3	0.4	0.2	0.32	0.25
c	0.35	0.2	0.4	0.45	0.2	0.4
d	0.2	0.4	0.2	0.2	0.4	0.2
e	0.4	0.4	0.2	0.05	0.3	0.3
f	0.4	0.2	0.1	0.2	0.4	0.2

4. The lookahead controller

A one-step lookahead controller was also implemented to control the two-tank system and the DC-DC converter. The

Table 3. Fitness obtained on the training cases.

Case	FSA	Lookahead
a	23655.5	6313.70
b	40863.8	55102.14
c	48887.8	32965.03
d	24290.5	8887.00
e	24351.8	22476.39
f	22375.5	20479.45

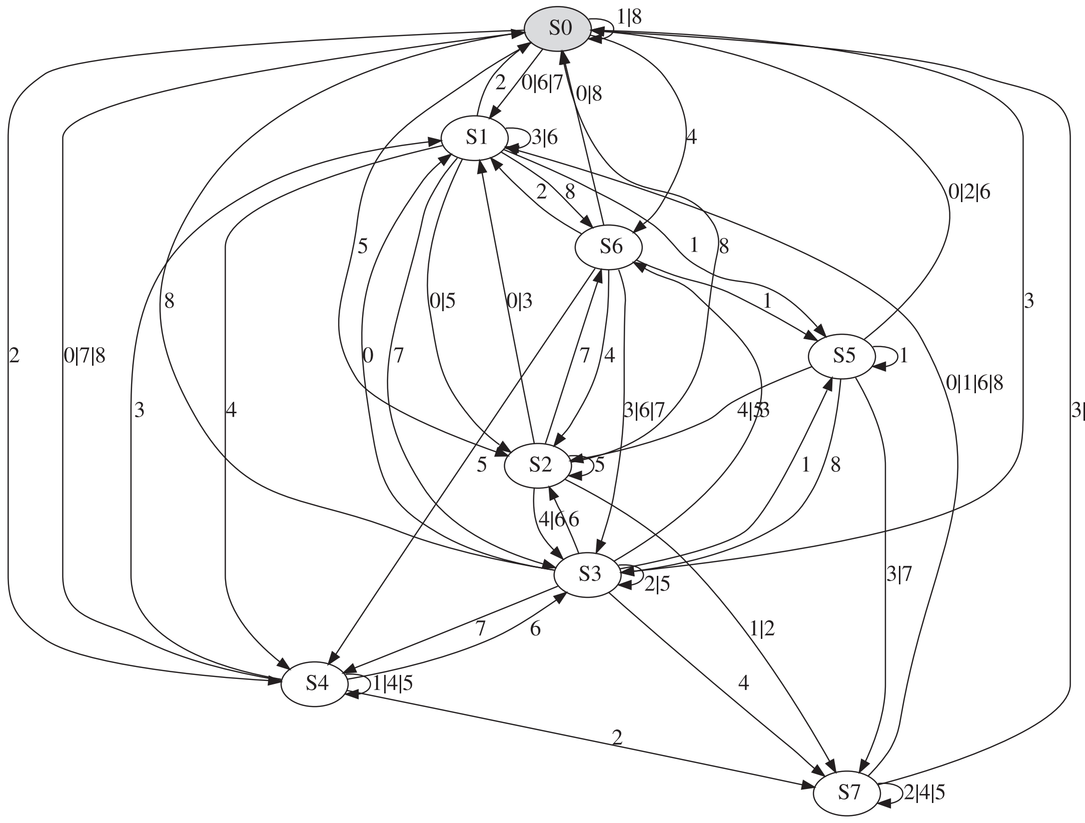


Figure 7. FSA controller for the two-tank system obtained at the end of the evolution.

one-step lookahead controller was chosen due to its simplicity, as well as reasonable computing expense in real-time execution. 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, $SW = \{Sw_i | i = 1, \dots, n\}$. Then, the chosen configuration to apply for the next time step will be the one having a direction most closely aligned with that to the target point x_d. Thereby, the controller minimises 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}:

$$\underset{Sw_i \in SW}{argmin} \left(\arccos \frac{(x - x_d)(x - x_{Sw_i})}{|x - x_d||x - x_{Sw_i}|} \right). \tag{9}$$

An important characteristic to note about this type of controller is that it 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 is the lookahead horizon, which should be established with regard to the available computing power and sampling time between two adjacent controller commands when the controller is in service. In our study, the lookahead horizon was set to be one step, in order to minimise computation time.

The fitnesses obtained by the lookahead controller on the training cases are listed in Table 3. It can be seen from Figure 11 that the lookahead controller failed to perform well at reaching its targets on the second tank in the first three cases, as seen in Figures 11 (f), 11 (d) and 11 (c). In these cases, the desired target asks for a level in tank 2 higher than that in tank 1. From the inspection of Figure 2, the only way to raise the level in the second tank is to first raise the level in tank 1 and then transfer the fluid from tank 1 to tank 2. Consequently, the level in tank 1 needs to go away from its target in order to help tank 2 reach its target. However, the one-step lookahead controller cannot generate an action sequence to achieve this target.

Figure 13 reveals the evolution of the two tank levels for the case of Figure 11 (d) under the supervision of both FSA and lookahead controllers. The dashed and solid lines are associated to the FSA and lookahead controllers, respectively. From Figure 13, we can see that when the system tried to make the transition from the first target t_1 to the second target t_2, there was a bifurcation between the two control trajectories at point a. There, the lookahead controller was not able to generate a control policy that would lead to point b, but instead led the trajectory toward point c, since

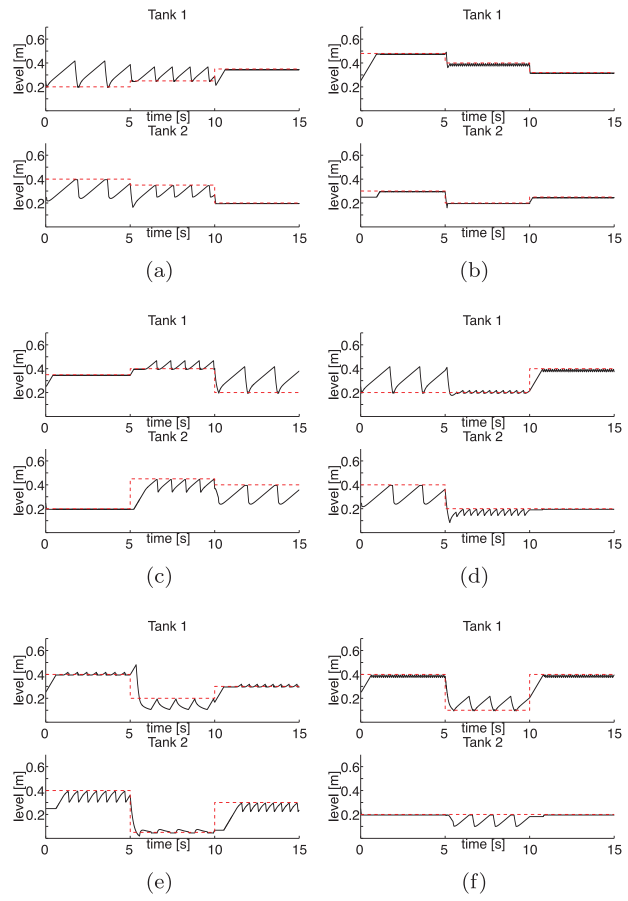


Figure 8. FSA controller behaviour on the training set for the two-tanks system of the best individual at the 150th generation.

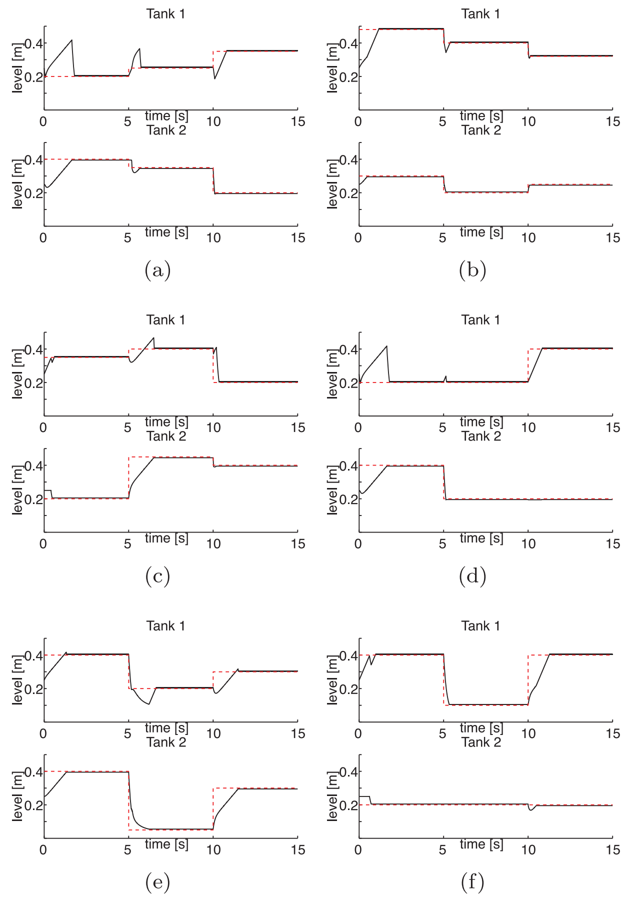


Figure 9. FSA controller behaviour on the training set of the two-tanks system. The tanks levels and desired targets are represented by a solid and dashed line, respectively.

it was more in the direction of the target t_2. However, once at c, the trajectory was pushed toward the third target t_3 and then missed target t_2, without being able to raise the level of the second tank.

In order to achieve target t_2, the controller would require a longer lookahead horizon in order to establish the correct sequence of action showing the payback of leaving the achieved level in the first tank. But when dealing with a system exhibiting a small time constant, the amount of simulation time required to go through a deep tree of possible states can easily become too long for the system to make a decision fast enough. Another problem is, depending on the complexity level of the control task, it is not obvious in advance what the correct lookahead horizon should be to accomplish all possible control targets.

5. Discussion

5.1. Capability of evolutionary design

Even though simultaneously reducing error in both tanks is not rewarded directly by the fitness function, the evolutionary algorithm was able to find an FSA controller providing the correct control sequence to meet the targets of both tanks. The result shows that the control sequence can drive the system away from the targets temporarily to make the necessary compromise to achieve the overall optimal performance. The results demonstrate the capability of the evolutionary design approach to achieve controllers satisfying difficult control tasks.

5.2. Issue of optimal design of FSA controllers

It is noteworthy that even though our evolved FSA controller has achieved apparent success in the case study, we can still not consider them as globally optimal solutions. This is because there are generally two aspects in FSA controller design, namely, the actuator control (control sequence generator) and the symbol generator, as indicated in Figure 1 (Koutsoukos et al. 2000). By its nature, the evolved FSA controller can only react when a predefined surface in the state space is crossed. In our project, we chose to use some very naive surfaces, basically defined based on the positions of targets in the hyperspace, as well as corresponding

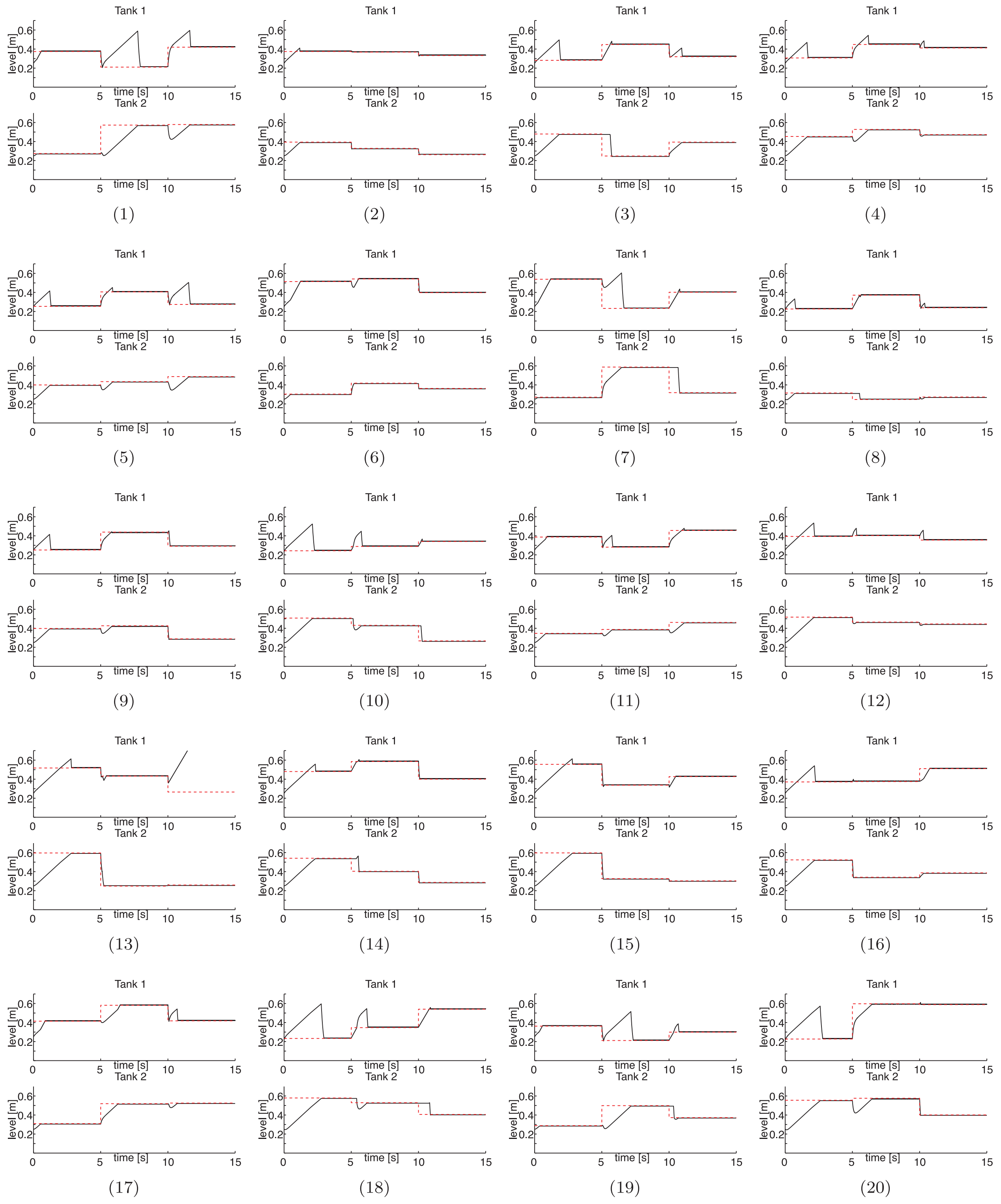


Figure 10. The behaviour of the evolved FSA controller on the testing set.

tolerance ranges to attract the trajectory toward the targets. However, these predefined surfaces are not necessarily the optimal ones. To illustrate this, let us consider a simple example of the double integrator system. The system is described in state-space representation as follows:

$$\dot{\mathbf{x}(t)} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u}(t), \tag{10}$$

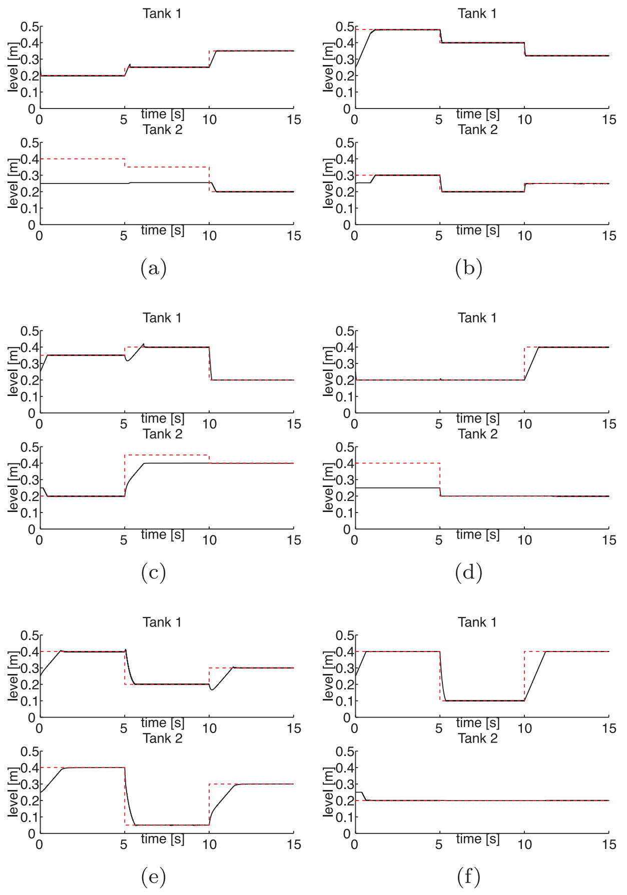


Figure 11. The lookahead controller behaviour on the target set of the two-tanks system. The tanks levels and desired targets are represented by a solid and dashed line, respectively.

where the control policies are

$$u(t) \in \{-1, 0, 1\}. \tag{11}$$

In Figure 14, we can see three different partitionings of the state space in order to generate symbols. First, the optimal partitioning is obtained using the invariant analysis

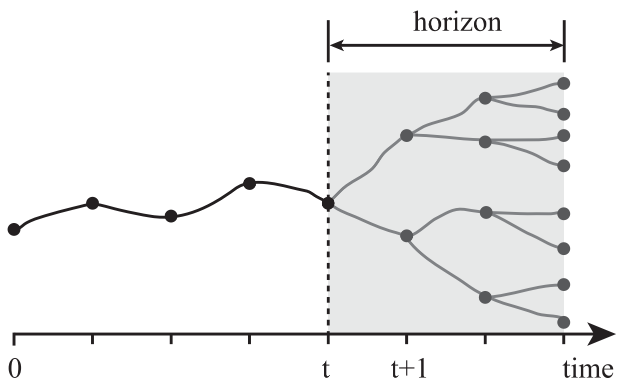


Figure 12. Lookahead control with limited horizon.

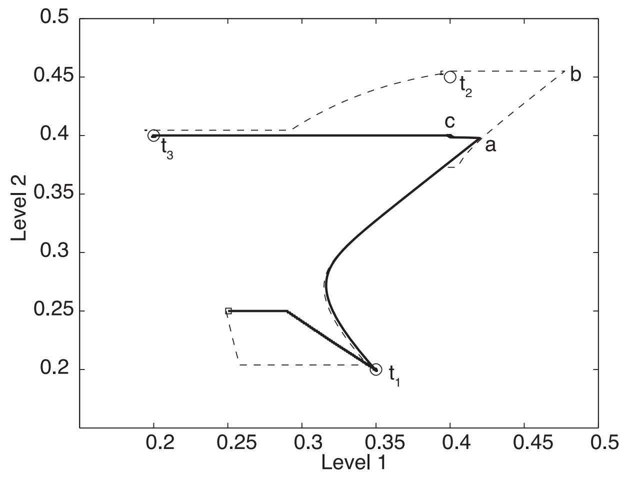


Figure 13. State-space trajectory for the two-tank system exhibiting failure of the lookahead controller. The FSA controller and the lookahead controller are represented by a dashed line and a solid line, respectively. The targets are marked by circles and the initial state is marked by a square.

described in Koutsoukos et al. (2000), which is marked by solid lines in the figure. Then, in Figure 14 (a), dashed lines mark the partitioning used by the experiment reported in this paper. It can be seen how poorly this partitioning is aligned to the optimal one. The partitioning is not good at grasping the specific dynamic of this system, because the input symbols may be generated when there is no relevant change in the vector field.

On the other hand, the partitioning could be better aligned to the optimal one if the surfaces are rotated by an angle as shown in Figure 14 (b). That way, we would have a chance of obtaining a better performing FSA, as the input symbols generated can better reflect relevant vector field changes. It is, therefore, a promising research pursuit to optimise the FSA controller in terms of both symbol generation and control sequence generation. Some preliminary work has been done in this direction, revealing that involving symbol generation in the evolution process can greatly increase the difficulty for the algorithm to converge to meaningful results. The search space seems to become so large that with a reasonably large computing resource as was used in work reported here, even some moderately performing FSA controllers could not be evolved. Further research in this direction should be done in the future.

5.3. *Potential of lookahead controller*

A lookahead controller was implemented so that a comparison study between the evolved FSA controller and the lookahead controller could be carried out (Figure 12). The experimental results demonstrate that the evolved FSA controller could achieve all six control targets used in the training stage, but that the lookahead controller failed in three

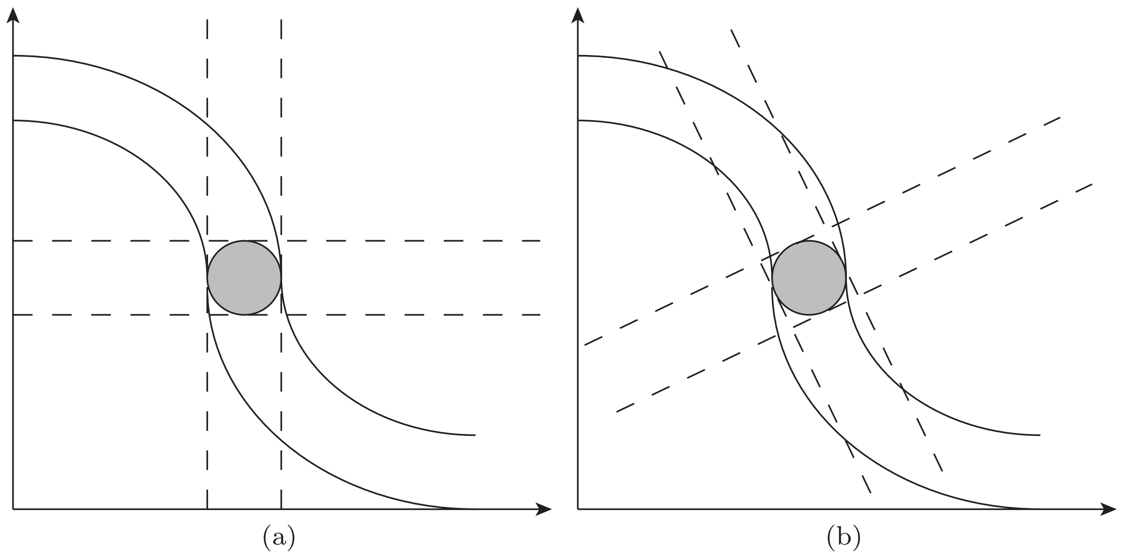


Figure 14. Double integrator partitioning. The circle represents the target and the solid lines are the optimal partitioning (Koutsoukos et al. 2000), while the dashed line shows a partitioning based solely on the target.

of them (when the water level of the second tank was asked to be higher than that of the first tank). In addition, the real-time execution speed of the lookahead controller can give rise to the issue of applicability in real-world situations, since the simulation time required to process a lookahead tree of possible states can become too long, even for fast systems.

With all these downsides, the lookahead controller still remains as a very useful method to apply because it follows a very forthright design procedure. Our study shows that some plant topologies are very amenable to the design procedure of lookahead control, but some others are not. For example, in the original topology of the two-tank system, the only way to fill the second tank is through the switch between the first and the second tanks, which makes it difficult to satisfy the target of the second tank without sacrificing that of the first one when the water level of the second tank is targeted to be higher than that of the first tank. However, if we modify the control plant by, for example, simply adding a pump directly to the second tank, the modified plant will then become very amenable to lookahead controller design. It is, therefore, very interesting to ask the following question: is it possible to explore the variations of control plants, so that we can find a control plant that can best match a controller design in terms of optimising the performance of the overall control system?

The answer seems to be positive. Even though most controller design still follows a sequential design procedure – i.e., we first fix the plant, and then try to find the best controller for the control plant – a real mechatronic system design philosophy supports concurrent design. That is, we should also allow the plant to be variable, and try to find a best match of the controller to the plant.

While it is reasonable to co-evolve both FSA controllers and hybrid bond graph structures which can in combination represent the complete hybrid system, the lookahead controller can be a better choice to represent the controller part. This is because for each candidate of plant, there exists a myriad of possibly good FSA controllers to be explored by evolutionary computation to locate a best one for the given plant. Because we also need to concurrently explore the space of the plants, the resulting work is an exploration of the product of space of controllers and space of plants, which may often be prohibitively large, if not impossible. In addition, the evolution can waste too much time in optimising the controller for a poor candidate of plant without promise.

In contrast, if we select a lookahead controller for the controller design, then we can avoid the situation of exploring in a product of spaces, as encountered in the previous approach. This is because for all candidates of plants, the design of a lookahead controller follows the same forthright procedure. In the condition that the lookahead horizon is predefined, each candidate of plant corresponds to a single lookahead controller. As a result, we only need to explore the design space of plants, and are relieved of all the work of exploring the other space of controllers. More details of this thrust of research will be presented in the future.

6. Conclusion

From the experimental results, we can see that the proposed approach that combines a genetic algorithm to explore the design space of FSA controllers with a hybrid bond graph to represent a hybrid mechatronic system can successfully design controllers for given plants to meet predefined control targets. The case study of the two-tank system design also demonstrates that, at least in this case, the evolved FSA controller generalises well to achieve different control targets than those used for training in the evolutionary process.

Even though the evolved FSA is complex, the process of interpreting the rules is very forthright. In addition, the interpreted individual rules are clear and simple. With enough motivation, one could eventually reverse engineer the set of rules used in the FSA. In contrast, interpreting a multilayered neural network can be much more challenging, because there exists no systematic way of interpretation.

To obtain an optimised FSA controller using an evolutionary approach requires significant computational power. However, the process of evolving/optimising the FSA controller is done offline. Once the optimal FSA controller is obtained, the controller can be easily implemented online, in an embedded system, for example, with no difficulty. The evolved FSA controller may appear to be complex, but its implementation and online execution will not require much computing effort and resources. It can, therefore, be applied very efficiently in the deployment stage.

On the other hand, the lookahead controller does not require any prior offline training or optimisation, but needs to repeatedly integrate the system model online, which drastically increases the computation power requirement at runtime. However, the lack of expensive training makes the lookahead controller an attractive controller representation, especially for exploring the open-ended design space of plants to achieve a further optimised performance of the overall hybrid mechatronic systems (Dupuis, Fan, and Goodman 2012).

While both approaches can be computationally intensive, the computational cost is simply moved between offline and online, depending on the architecture chosen.

Through the above analysis of this comparative study, more insight is obtained which can be summarised in the following conclusions:

(1) When the plant is modifiable, it is more suitable to use a lookahead controller as the representation of the controller. If, instead, the Finite State Machine (FSM) is used as the representation of the controller, the search space becomes so prohibitively large that it prevents the evolutionary algorithm from achieving a reasonably good result in a limited timeframe, at least using the methods described here.
(2) When the plant is given and fixed, it is more suitable to use an FSM as the representation of the controller. Even though it takes more time to evolve an optimised FSM controller offline, the online execution time of the FSM controller is much shorter than that of the lookahead controller. In addition, the performance of the lookahead controller is dependent on the given plant – i.e., for a certain given plant, it is difficult for the lookahead controller to always achieve good performance, as is demonstrated by the case study in this paper.

It is, therefore, recommended that the following pragmatic steps be taken in a systematic procedure for evolutionary design of both controller and plant of a hybrid dynamic system.

(1) First, use a lookahead controller as the controller representation and allow both the plant and the controller to be modified in a concurrent evolutionary design stage. As outputs of this stage, an optimised plant is obtained along with its lookahead controller.
(2) Then use the optimised plant evolved in the first stage as the given plant, and using an FSM representation for the controller, evolve a new controller for that given plant.
(3) Compare the performance of the FSM controller with that of the lookahead controller. If the performance of the FSM is not worse than that of the lookahead controller, use the FSM controller. Otherwise, if the lookahead controller satisfies the requirements for online execution time for the application, select it as the controller.

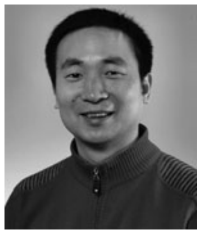
An important step for future research is to apply this approach in a real-world application, in which the plant to be controlled is often complex and its dynamics are not known in advance. To obtain a model so that we can apply the approach, it may be advisable to use surrogate models (Jin 2005; Lim, Jin, Ong, and Sendhoff 2010) to represent the plant, which can also help to cope with uncertainty and noise in real-world applications. In the current application, only the topology information of the finite state automaton is encoded. In real-world applications, it is of great interest to extend the approach so that both parameters and topology can be optimised simultaneously in the evolutionary design process. In this situation, memetic computing (Ong, Lim, and Chen 2010; Chen, Ong, Lim, and Tan 2011) can be a very suitable approach, in which local search methods can be applied and can focus more on parameter optimisation, while evolutionary search focuses more on topology exploration.

Acknowledgements

This work was supported in part by the Leading Talent Project of Guangdong Province, in part by the National Science Foundation under Cooperative Agreement DBI-0939454, in part by the Chinese National High-Tech 863 Project under Grant 2013AA100305 and in part by the National Natural Science Foundation of China under Grant 61175073 and 61174090.

Notes on contributors

Jean-François Dupuis received a BEng degree in Mechanical Engineering and a MS in Electrical Engineering from Laval University, Québec, Canada in 2003 and 2007, respectively, and a PhD degree in Mechanical Engineering from the Technical University of Denmark in 2011.



Zhun Fan received the BSc and MSc degrees in Control Engineering from Huazhong University of Science and Technology, Wuhan, China, in 1995 and 2000, respectively, and the PhD degree in electrical engineering from the Michigan State University, USA, in 2004.

From 2004 to 2007, he was an Assistant Professor at the Technical University of Denmark. From 2007 to 2011, he was an Associate Professor at the Technical University of Denmark. He is currently a Professor and Head of the Department of Electrical Engineering at Shantou University, Guangdong, China.

Dr. Fan has been principle investigator of various projects sponsored by Danish Research Agency of Science Technology and Innovation. His research is also supported by National Science Foundation, USA, and National Natural Science Foundation of China. His major research interests include evolutionary computation, intelligent control and robotic systems, robot vision and cognition, MEMS, design automation and optimisation, intelligent power system and transportation system, etc.

Dr. Fan is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE). He is also a member of ACM and ASME.

Erik D. Goodman is 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 and funded beginning in 2010. His research centres on application of evolutionary principles to solution of engineering design problems. He received the PhD in computer and communication sciences from the University of Michigan, Ann Arbor, in 1971. He became Asst. Prof. of Electrical Engineering and Systems Science in 1972, Assoc. Prof. in 1978 and Prof. in 1984, all at Michigan State University, 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 (GARAGe) since its founding in 1993. He is co-founder and vice president of Red Cedar Technology, Inc., a firm that develops design optimisation software for use in industry. He was chosen Michigan Distinguished Professor of the Year, 2009, by the Presidents Council, State Universities of Michigan.

Prof. Goodman was Chair of the Executive Board and a Senior Fellow of the International Society for Genetic and Evolutionary Computation, 2003–2005. He was founding chair of the ACM's Special Interest Group on Genetic and Evolutionary Computation (SIGEVO), serving from 2005 to 2007.

References

Ashlock, D. (2006), 'Evolving Finite State Automata', in *Evolutionary Computation for Modeling and Optimization*, New York: Springer, pp. 143–166.

Bemporad, A. (2003), 'Modeling, Control, and Reachability Analysis of Discrete-Time Hybrid Systems', Technical Report, University of Siena, Dept. of Information Engineering.

Benson, K. (2000), 'Evolving Finite State Machines With Embedded Genetic Programming for Automatic Target Detection', *Proceedings of the 2000 Congress on Evolutionary Computation*, 2, 1543–1549.

Buisson, J., Cormerais, H., and Richard, P.Y. (2002), 'Analysis of the Bond Graph Model of Hybrid Physical Systems With Ideal Switches', *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 216(1), 47–63.

Chen, X., Ong, Y.S., Lim, M.H., and Tan, K.C. (2011), 'A Multi-Facet Survey on Memetic Computation', *IEEE Transactions on Evolutionary Computation*, 15(5), 591–607.

Chen, W.N., Zhang, J., Chung, H., Zhong, W.L., Wu, W.G., and hui Shi, Y. (2010), 'A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems', *IEEE Transactions on Evolutionary Computation*, 14(2), 278–300.

Cormerais, H., Buisson, J., Richard, P., and Morvan, C. (2008), 'Modelling and Passivity Based Control of Switched Systems From Bond Graph Formalism: Application to Multicellular Converters', *Journal of the Franklin Institute*, 345(5), 468–488.

Dunay, B.D., Petry, F.E., and Buckles, B.P. (1994), 'Regular Language Induction With Genetic Programming', *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 1, 396–400.

Dupuis, J.F., Fan, Z., and Goodman, E.D. (2012), 'Evolutionary Design of Both Topologies and Parameters of a Hybrid Dynamical System', *IEEE Transactions on Evolutionary Computation*, 16(3), 391–405.

Fan, Z., Seo, K., Hu, J., Goodman, E.D., and Rosenberg, R.C. (2004a), 'A Novel Evolutionary Engineering Design Approach for Mixed-Domain Systems', *Engineering Optimization*, 36(2), 127–147.

Fan, Z., Wang, J., Achiche, S., Goodman, E., and Rosenberg, R. (2008), 'Structured Synthesis of Mems Using Evolutionary Approaches', *Applied Soft Computing*, 8(1), 579–589.

Fan, Z., Wang, J., and Goodman, E. (2004b), 'Exploring Open-Ended Design Space of Mechatronic Systems', *International Journal of Advanced Robotic Systems*, 1, 295–302.

Fogel, L.J. (1962), 'Autonomous Automata', *Industrial Research*, 4, 14–19.

Gagné, C., and Parizeau, M. (2006), 'Genericity in Evolutionary Computation Software Tools: Principles and Case Study', *International Journal on Artificial Intelligence Tools, 15(2), 173–194.*

Jin, Y. (2005), 'A Comprehensive Survey of Fitness Approximation in Evolutionary Computation', *Soft Computing*, 9(1), 3–12.

Karnopp, D., Margolis, D., and Rosenberg, R. (2005), *System Dynamics: Modeling and Simulation of Mechatronic Systems* (4th ed.), Hoboken, NJ: John Wiley and Sons.

Koutsoukos, X.D., Antsaklis, P.J., Stiver, J.A., and Lemmon, M.D. (2000), 'Supervisory Control of Hybrid Systems', *Proceedings of the IEEE*, 88, 1026–1049.

Kung, Y.S., Huang, C.C., and Tsai, M.H. (2009), 'FPGA Realization of an Adaptive Fuzzy Controller for PMLSM Drive', *IEEE Transactions on Industrial Electronics*, 56(8), 2923–2932.

Li, Q., Zhang, W.J., and Chen, L. (2001), 'Design for Control – A Concurrent Engineering Approach for Mechatronic Systems Design', *IEEE/ASME Transactions on Mechatronics*, 6, 161–169.

Lim, D., Jin, Y., Ong, Y.S., and Sendhoff, B. (2010), 'Generalizing Surrogate-Assisted Evolutionary Computation', *IEEE Transactions on Evolutionary Computation*, 14(3), 329–355.

Mosterman, P.J. (1997), *Hybrid Dynamic Systems: A Hybrid Bond Graph Modeling Paradigm and its Application in Diagnosis*, Nashville: Vanderbilt University.

MPI (2008), *MPI: A Message-Passing Interface Standard*, Message Passing Interface Forum, http://www.mpi-forum.org.

Ong, Y.S., Lim, M.H., and Chen, X. (2010), 'Memetic Computation - Past, Present & Future [Research Frontier]', *IEEE Computational Intelligence Magazine*, 5(2), 24–31.

Ortega, R., Perez, J.A.L., Nicklasson, P.J., Sira-Ramirez, H.J., and Sira-Ramirez, H. (1998), *Passivity-based Control of Euler-Lagrange Systems: Mechanical, Electrical and Electromechanical Applications*, Communications and Control Engineering, London: Springer.

Ramadge, P.J., and Wonham, W.M. (1987), 'Supervisory Control of a Class of Discrete Event Processes', *SIAM Journal on Control and Optimization*, 25(1), 206–230.

Sainz, M., Armengol, J., and Vehi, J. (2002), 'Fault Detection and Isolation of the Three-Tank System Using the Modal Interval Analysis', *Journal of Process Control*, 12(2), 325–338.

Samosir, A.S., and Yatim, A.H.M. (2010), 'Implementation of Dynamic Evolution Control of Bidirectional DC-DC Converter for Interfacing Ultracapacitor Energy Storage to Fuel-Cell System', *IEEE Transactions on Industrial Electronics*, 57(10), 3468–3473.

Seo, K., Fan, Z., Hu, J., Goodman, E.D., and Rosenberg, R.C. (2003), 'Toward A Unified and Automated Design Methodology for Multi-Domain Dynamic Systems Using Bond Graphs and Genetic Programming', *Mechatronics*, 13(8–9), 851–885.

Stanley, K.O., and Miikkulainen, R. (2002), 'Evolving Neural Networks Through Augmenting Topologies', *Evolutionary Computation*, 10(2), 99–127.

Stiver, J., Koutsoukos, X., and Antsaklis, P. (2000), 'An Invariant Based Approach to the Design of Hybrid Control Systems', Technical Report, University of Notre Dame.

Stiver, J., Koutsoukos, X., and Antsaklis, P. (2001), 'An Invariant Based Approach to the Design of Hybrid Control Systems', *International Journal of Robust and Nonlinear Control*, 11(5), 453–478.

Strömberg, J.E., Nadjm-Tehrani, S., and Top, J. (1996), 'Switched Bond Graphs as Front-End to Formal Verification of Hybrid Systems', *Hybrid Systems III*, 1066, 282–293.

Sun, Z., and Ge, S. (2005a), 'Analysis and Synthesis of Switched Linear Control Systems', *Automatica*, 41(2), 181–195.

Sun, Z., and Ge, S.S. (2005b), *Switched Linear Systems : Control and Design*, Communications and Control Engineering, London: Springer.

Wai, R.J., and Tu, C.H. (2007), 'Development of Lyapunov-Based Genetic Algorithm Control for Linear Piezoelectric Ceramic Motor Drive', *IEEE Transactions on Industrial Electronics*, 54(5), 2566–2582.

Wang, J., Fan, Z., Terpenny, J.P., and Goodman, E.D. (2005), 'Knowledge Interaction With Genetic Programming in Mechatronic Systems Design Using Bond Graphs', *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 35, 172–182.

Wang, J., Fan, Z., Terpenny, J.P., and Goodman, E.D. (2008), 'Co-operative Body-Brain Coevolutionary Synthesis of Mechatronic Systems', *AI EDAM*, 22(3), 219–234.

Wu, J., Biswas, G., Abdelwahed, S., and Manders, E. (2005), 'A Hybrid Control System Design and Implementation for a Three Tank Testbed', *IEEE Conference on Control Applications*, 645–650.

Yu, X., and Kaynak, O. (2009), 'Sliding-Mode Control With Soft Computing: A Survey', *IEEE Transactions on Industrial Electronics*, 56(9), 3275–3285.