

Improving the Quality of Supervised Finite-State Machine Construction Using Real-Valued Variables

Igor Buzhinsky, Daniil Chivilikhin, Vladimir Ulyantsev,
Fedor Tsarev



Master student (2013–2015)
ITMO University
Computer Technologies Laboratory
igor.buzhinsky@gmail.com

GECCO 2014, Student Workshop



July 12, 2014

We focus on finite-state machines (FSMs) for control tasks.

Why FSMs?

- ▶ Easy to comprehend and visualize
- ▶ Can be formally verified with the *Model Checking* approach

Where FSMs can be applied?

- ▶ Control systems: for energy, aircraft, space industries...
- ▶ Where reliability is important

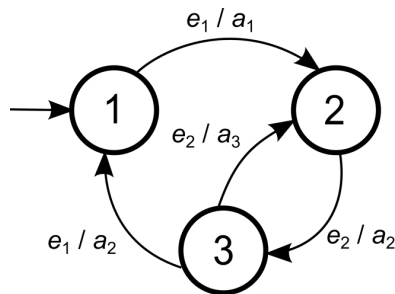
FSMs now:

- ▶ Quantum Leaps
- ▶ IEC 61499 standard for distributed PLC systems
- ▶ StateFlow for MATLAB

FSM: definition

FSM = $(S, s_0, E, A, \delta, \lambda)$

- ▶ S – finite set of states
- ▶ s_0 – initial state
- ▶ E – event set
- ▶ A – action set
- ▶ $\delta : S \times E \rightarrow S$ – transition function
- ▶ $\lambda : S \times E \rightarrow A$ – output function



Inducing FSMs from specification

Specification:

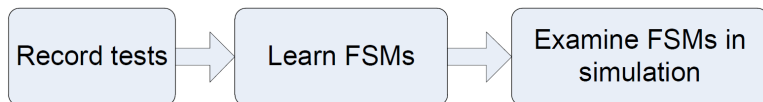
- ▶ Tests / scenarios, software traces
- ▶ Temporal properties (LTL, CTL formulae)
 - ▶ Example: $G((p \wedge \neg q) \rightarrow X r)$

FSMs:

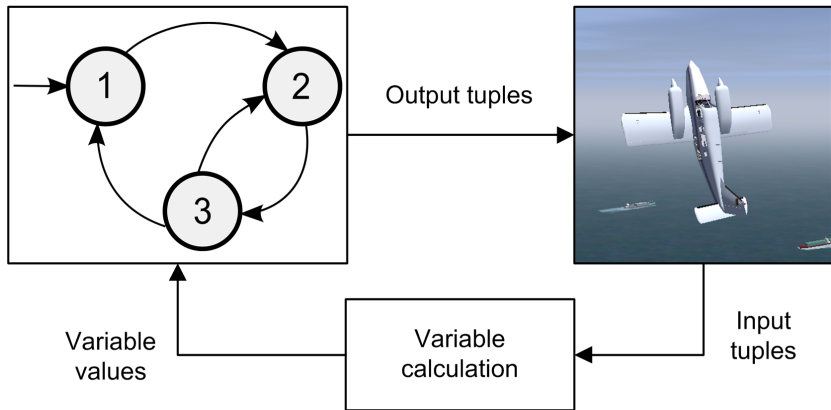
- ▶ Are software models
- ▶ Can be induced with metaheuristics (see Tsarev & Egorov, GECCO '11)
- ▶ Can be easily transformed into source code

Considered problem in brief

- ▶ Induce an FSM to control an object in a complex environment with **continuous** inputs and outputs
- ▶ $\text{in}[i, t]$, $\text{out}[i, t]$ – training data ($i = 1..N$ is a number of a test, $t = 1..\text{len}[i]$ is a timestamp)
- ▶ The **aircraft control** problem is considered as an example (*FlightGear* simulator is used)
- ▶ Inputs correspond to sensor values, outputs correspond to control device positions
- ▶ Methodology:



FSM and aircraft interaction

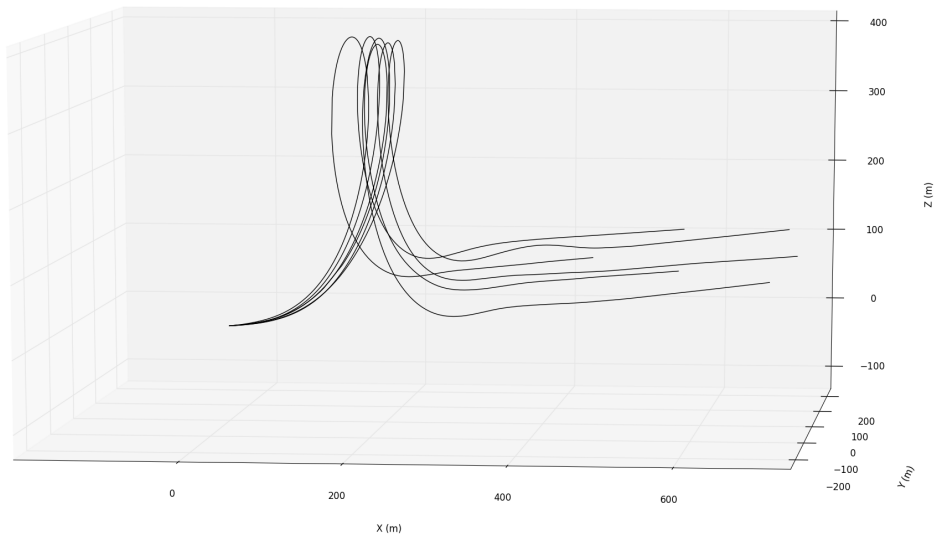


Test example

Values	Description	$t = 1$...	$t = 10$...
$\text{in}[i, t]_1$	Pitch angle ($^\circ$)	3.078	...	3.544	...
$\text{in}[i, t]_2$	Roll angle ($^\circ$)	-0.076	...	0.351	...
$\text{in}[i, t]_3$	Heading ($^\circ$)	198.03	...	198.11	...
$\text{in}[i, t]_4$	Airspeed (knots)	251.42	...	252.29	...
$\text{out}[i, t]_1$	Aileron position	0.000	...	0.032	...
$\text{out}[i, t]_2$	Rudder position	0.000	...	0.016	...
$\text{out}[i, t]_3$	Elevator position	-0.035	...	-0.039	...

Test example (4 inputs, 3 outputs)

Aircraft path examples (loop)



- ▶ A. Alexandrov, A. Sergushichev, S. Kazakov, F. Tsarev. Genetic algorithm for induction of finite automata with continuous and discrete output actions. GECCO '11 Companion, pp. 775–778. ACM, 2011.
 - ▶ GA, several hours to construct an FSM
 - ▶ Inputs are transformed to **predicate** values
 - ▶ Several transitions per time step (for each predicate)
 - ▶ Automatic output derivation
- ▶ I. Buzhinsky, V. Ulyantsev, A. Shalyto. Test-based induction of finite-state machines with continuous output actions. Proceedings of MIM '13, pp. 1049–1054. IFAC, 2013.
 - ▶ ACO, about 10 minutes to construct an FSM

Proposed approach (outline)

Predicates for transitions:

- ▶ Example: $p(\text{input}) = (\text{input}_2 < 3.6)$
- ▶ One transition per time step
- ▶ Only few (“significant”) predicates are used in each state
- ▶ Example of a transition table:

Condition	$\neg p_1 \wedge p_3$	$\neg p_1 \wedge p_3$	$p_1 \wedge \neg p_3$	$p_1 \wedge p_3$
New state	2	4	1	1

Real-valued **variables** for output actions:

- ▶ Example: $v(\text{input}) = (\text{input}_1 - 0.5)^2$
- ▶ Action update: $u'_i = u_i + \sum_{j=1}^k r_{s,i,j} v_j$, where $r_{s,i,j}$ are constant for a fixed FSM

- ▶ Initial state

For each state:

- ▶ Mask of predicate significance
- ▶ Transition table for all combinations of values of significant predicates
- ▶ Mask of variable significance for each control device (output)

Actions (values $r_{s,i,j}$) **are not included in individuals, but are derived** using a procedure which reminds solving normal equation for linear regression. Linearity of the action update is important here.

FSM example (one state)

Upper box: included in an individual

Predicate significance mask			
p_1	p_2	p_3	p_4
✓	✗	✓	✗

Transition table (from the current state)			
$\neg p_1 \wedge \neg p_3$	$\neg p_1 \wedge p_3$	$p_1 \wedge \neg p_3$	$p_1 \wedge p_3$
2	4	1	1

Variable significance mask for output 1			
$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$v_{1,4}$
✗	✓	✗	✓

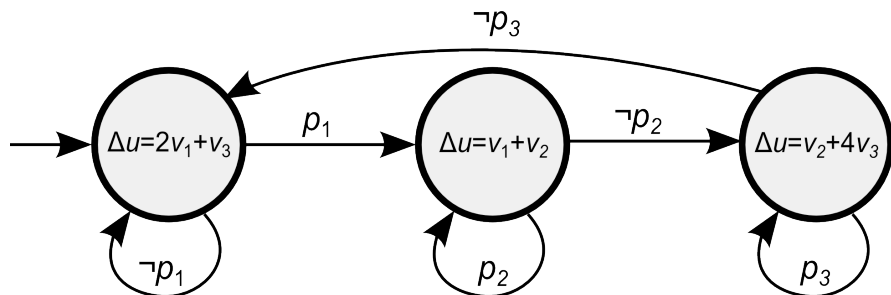
Variable significance mask for output 2				
$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$v_{2,4}$	$v_{2,5}$
✓	✗	✗	✗	✓

Output action 1			
$r_{s,1,1}$	$r_{s,2,1}$	$r_{s,3,1}$	$r_{s,4,1}$
0	1.2	0	0.3
$\Delta u_1 = 1.2 v_{1,2} + 0.3 v_{1,4}$			

Output action 2				
$r_{s,1,2}$	$r_{s,2,2}$	$r_{s,3,2}$	$r_{s,4,2}$	$r_{s,5,2}$
3.7	0	0	0	-0.3
$\Delta u_2 = 3.7 v_{2,1} - 0.3 v_{2,5}$				

Lower box: derived for each individual

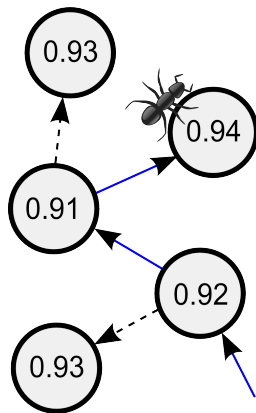
FSM example (transition graph)



- ▶ Example for 3 predicates, 3 variables, and one output u
- ▶ Similar masks and actions in each state

- ▶ Fitness function: $f = 1 - P_\rho - P_\tau$
- ▶ P_ρ : penalty for the distances between the reference outputs ($\text{out}[i, t]$) and the outputs produced by the individual, executed on tests
- ▶ P_τ : penalty for **state changes** (an FSM with clearly distinct states is better)
- ▶ f is maximized for each individual

- ▶ ACO-based algorithm
- ▶ D. Chivilikhin, V. Ulyantsev. MuACOsm – a new mutation-based ant colony optimization algorithm for learning finite-state machines. Proceedings of GECCO '13, pp. 511–518. ACM, 2013.
- ▶ Uses only mutations
- ▶ Simplification: pheromone removed
- ▶ Parameters were tuned with *irace*



Experimental setup

- ▶ Proposed FSM representation (induced with the modified fitness function) vs. Alexandrov et al.
- ▶ 50 ACO executions for each combination of the FSM representation, number of states $|S| = 3, 4, 5$ and test set (loop, barrel roll, turn)
- ▶ Termination criterion: stagnation after 5000 fitness evaluations
- ▶ About 10 minutes for each execution on a quad-core computer

- ▶ Individual with maximum value of f for each execution was run in simulation 10 times
- ▶ Roll and pitch quality metrics were compared across the representations

Results: fitness values

$ S $	FSM Representation	Loop	Barrel roll	Turn
3	Proposed	0.9856	0.9854	0.9892
	Previous	0.9812	0.9832	0.9894
4	Proposed	0.9866	0.9863	0.9898
	Previous	0.9836	0.9856	0.9901
5	Proposed	0.9873	0.9868	0.9901
	Previous	0.9842	0.9858	0.9902

Median fitness values for different test sets and number of states

Results: simulation

$ S $	FSM Representation	Loop	Barrel roll	Turn
3	Proposed	1.71/17.21	16.52/3.20	4.80/1.95
	Previous	6.37/20.54	18.56/4.44	50.29/7.58
4	Proposed	2.41/23.04	15.35/2.51	4.10/1.42
	Previous	6.32/22.11	21.86/4.08	57.04/6.79
5	Proposed	3.21/25.27	14.74/2.43	4.07/1.36
	Previous	9.54/24.44	22.99/4.68	45.83/7.83

Median roll/pitch errors ($^{\circ}$) for different test sets and number of states

A screenshot (loop, FlightGear simulator)



A screenshot (180° turn, FlightGear simulator)



- ▶ A new representation method for FSMs with continuous inputs and outputs
- ▶ Automatic output derivation for the new representation
- ▶ A known fitness function was modified to get more comprehensible FSMs
- ▶ Simulation quality of FSMs was improved

- ▶ Automatic construction of predicates and variables
- ▶ Testing the approach in different environments (e.g. robot simulators)
- ▶ Involving more types of specification (including temporal properties)



Improving the Quality of Supervised Finite-State Machine Construction Using Real-Valued Variables

Igor Buzhinsky, Daniil Chivilikhin, Vladimir Ulyantsev, Fedor Tsarev
ITMO University, St. Petersburg
Computer Technologies Laboratory



Problem Statement

- Finite-State Machine:
 - FSM = $(S, s_0, E, A, \delta, \lambda)$
 - S - finite set of states
 - s_0 - initial state
 - E - event and action sets
 - $\delta: S \times E \rightarrow S$ - transition function
 - $\lambda: S \times E \rightarrow A$ - output function
- Real-valued inputs and outputs
- Tests ($N = 20-30$) are the examples of proper control
- Construct an FSM with behavior close to the tests
- Accuracy model is used as a controlled object
- Tests can be written manually in a flight simulator
- Test example:

Index	Description	F = 1	F = 10	F = 2.95
1	Pitch angle (°)	5.078	5.544	4.312
2	Amplitude (msec)	251.42	252.28	253.20
3	Ammon position	0.030	0.032	0.031
4	Elevator position	-0.035	-0.039	-0.037



FSM Learning

Methodology: Record tests \rightarrow Learn FSMs \rightarrow Examine FSMs in simulation

- ACO-based algorithm:
 - FSMs with unrolled output functions are individuals
 - Output actions are derived so that the fitness function is maximized
 - Fitness function:

$$f = 1 - \sqrt{\frac{1}{N} \sum_{i=1}^N |p(\text{test}[i], \text{act})|} - R - \sqrt{\frac{1}{N} \sum_{i=1}^N |\text{max}(|s_i| - |S| + 1.0|)^2}$$
- $$p(\text{test}[i], \text{act}[j]) = \frac{1}{|\text{test}[i]|} \sum_{k=1}^{|\text{test}[i]|} \sum_{l=1}^{|\text{act}[j]|} \frac{\text{act}[l, i] \cdot \text{test}[k, i]}{\sigma_{\text{act}[j]}^2 + \sigma_{\text{test}[i]}^2}$$
- C - number of outputs
- act(j) - FSM's output for the j-th test
- x - number of state changes on the i-th test

FSM Representation

Refined tables for transition function

Output variables are linear

x_1	x_2	y_1	y_2
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0

Predictors (Boolean)

Boolean inputs define variables are independent of output generation

Real-valued variables (arbitrary transitions) are used as input variables for different states

More complex example (five states)

Transition digital table	Transition table (the current state)																														
<table border="1"> <tr> <th>State</th> <th>x_1</th> <th>x_2</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>2</td> <td>1</td> <td>0</td> </tr> <tr> <td>3</td> <td>1</td> <td>1</td> </tr> </table>	State	x_1	x_2	0	0	0	1	0	1	2	1	0	3	1	1	<table border="1"> <tr> <th>State</th> <th>y_1</th> <th>y_2</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>2</td> <td>0</td> <td>0</td> </tr> <tr> <td>3</td> <td>0</td> <td>0</td> </tr> </table>	State	y_1	y_2	0	0	0	1	0	0	2	0	0	3	0	0
State	x_1	x_2																													
0	0	0																													
1	0	1																													
2	1	0																													
3	1	1																													
State	y_1	y_2																													
0	0	0																													
1	0	0																													
2	0	0																													
3	0	0																													

Experiments & Results

- Quad-core Intel Core i7-2670QM processor
- 3 test sets, number |S| of states 3-5
- Method execution time: about 10 minutes
- Comparison with the previous representation
- Fitness values:

S	FSM Representation	Loop	Rand cell	Turn
3	Proposed	0.9566	0.9854	0.9922
3	Previous	0.9822	0.9812	0.9894
4	Proposed	0.9666	0.9863	0.9668
4	Previous	0.9438	0.9358	0.9301
5	Proposed	0.9873	0.9860	0.9901
5	Previous	0.9642	0.9358	0.9302
- Reliability errors (%) in simulation:

S	FSM Representation	Loop	Rand cell	Turn
3	Proposed	1.73/17.21	16.52/3.20	4.80/3.95
3	Previous	6.37/23.34	15.96/4.44	15.35/7.53
4	Proposed	2.41/23.84	15.35/3.51	4.10/3.43
4	Previous	6.32/22.11	23.96/4.28	12.04/6.79
5	Proposed	3.23/22.97	14.74/3.43	4.62/3.36
5	Previous	9.54/24.44	12.96/4.68	15.83/7.83
- Quality is improved
- Now it is possible to construct FSMs performing the turn

Screenshots (FlightGear simulator)

Loop Turn

Publications

- Buzhinsky I., Ulyantsev V., Chivilikhin D., Struts A. Inferring Finite State Machine from Training Samples Using Ant Colony Optimization. Journal of Control and System Science International, 2014, Vol. 10, No. 2, P. 268-286.
- Buzhinsky I., Ulyantsev V., Tsarev F., Struts A. Search-Based Construction of Finite State Machine from Real-Valued Actions. New Representation Model, Models and Evolutionary Computation Conference (NEOCC) 2013, Conference, P. 189-203.
- buzhinsky_chivilikhin_ulyantsev@raim.ifmo.ru
- <http://raim.ifmo.ru/en/87645/>



Inferring Automata-Based Programs from Specification With Mutation-Based Ant Colony Optimization

Daniil Chivilikhin and Vladimir Ulyantsev
ITMO University
Computer Technologies Laboratory
Saint Petersburg, Russia



Reliable control system development

Traditional approach vs. Automata-based programming

Inferring automata-based programs

Test indicators \rightarrow Automata-based program inference \rightarrow Genetic algorithm \rightarrow Construct automata program

Example: elevator doors control

Goals:

- Open "open closed" door
- Close "open closed" door
- Close "closed opened" door
- Open "closed opened" door
- Close "closed opened" door
- Open "closed opened" door

Actions:

- Close "open closed" door
- Open "open closed" door
- Close "closed opened" door
- Open "closed opened" door
- Close "closed opened" door
- Open "closed opened" door

Mutation-Based Ant Colony Optimization

Mutation graph

- Nodes - Finite-State Machine
- Edges - FSM mutations

Algorithm

- Initialize (Start)
- For each iteration with ant colony
 - For each ant
 - For each transition with ant colony
 - Check also updates
- Pheromone update
- Check also updates
- Update τ_{ij}

Fitness function

$f(x) = \frac{1}{N} \sum_{i=1}^N \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} \frac{1}{\sigma_{S_i}^2 + \sigma_{act}^2} \sum_{k=1}^{|act} \frac{act[k, i] \cdot test[j, i]}{\sigma_{act}^2 + \sigma_{test}^2}$

Empirical study

Equipment

- GA-based MUACO with genetically used GA
- GA-based solver with 100% success in solving using the state
- Q2 LTL formulae for each algorithm
- Q2 LTL formulae for each instance

Setup

- $GA_{pop} = 4 \cdot 10^4$
- Q2 formulae length = 100-150
- Q2 instances for each value of popsize

Publications

- Chivilikhin D., Ulyantsev V. MUACO - A New Mutation-Based Ant Colony Optimization Algorithm for Learning Finite-State Machine. In Proceedings of the 11th Genetic and Evolutionary Computation Conference (GECCO'13), Christian Bunn (Ed.), ACM, New York, NY, USA, 2013, pp. 511-518
- Ulyantsev V., Tsarev F. Extended Finite-State Machine Induction using SAT-Solver / Proceedings of the 14th IFAC Symposium "Information Control Problems in Manufacturing - INCOM'12", SPb, 2012, pp. 516-517

Acknowledgements

This work was financially supported by:

- the Government of Russian Federation, Grant 074-U140;
- RFBR, research project No. 14-07-91337 mol_a

chivilikhin_ulyantsev@raim.ifmo.ru
http://raim.ifmo.ru/~chivilikhin_ulyantsev/
 GitHub site: <http://raim.ifmo.ru/en/87645/>

Thank you for your attention!
Any questions?

Improving the Quality of Supervised Finite-State Machine Construction
Using Real-Valued Variables

Igor Buzhinsky, Daniil Chivilikhin, Vladimir Ulyantsev, Fedor Tsarev
`igor.buzhinsky@gmail.com`

This presentation online:
<http://goo.gl/k4yCqG>