Condition–action rules in controlling complex systems

Sotiris Moschoyiannis
joint work with
Matthew R. Karlsen and Vlad Georgiev

University of Surrey, England

RuleML Webinar @ Skype

29th March 2019

Outline

Introduction

OJPA Project

Onward Journey Planning Assistant

Personalised recommendations for:

- customers planning a journey
- customers experiencing disruption

Overall challenge

- Input:
  - environment factors:
    - train, taxi, tube, boat and bus [**0 or 5**]; weather [**0 to 5**]
  - journey-specific factors:
    - current delay, delay on current mode(s), onward delay [**0 or 5**]
  - passenger preferences:
    - value, speed, comfort, shelter [**0 to 5**]
- Output:
  - "Correct" single integer recommendation:
    - no change (**0**), single taxi (**1**), shared taxi (**2**), bus (**3**), boat (**4**), tube (**5**), train (**6**)

Population of Rules (the Knowledge)

Condition	:	Action
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [0,1] [#,#] [2,5] [#,#] [#,#]	:	1
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [0,1] [2,5] [#,#] [#,#] [#,#]	:	1
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [2,3] [#,#] [4,5] [#,#] [#,#]	:	1
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [2,3] [4,5] [#,#] [#,#] [#,#]	:	1
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [0,1] [0,1] [2,3] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [0,1] [2,3] [0,1] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [0,1] [2,3] [2,3] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [2,3] [0,1] [2,3] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [2,3] [2,3] [2,3] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [0,1] [2,3] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [0,1] [4,5] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [2,3] [2,3] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [2,3] [4,5] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [4,5] [0,1] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [4,5] [2,3] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [4,5] [4,5] [#,#] [#,#]	:	2
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [0,1] [0,1] [0,1] [#,#] [#,#]	:	3
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [2,3] [0,1] [0,1] [#,#] [#,#]	:	3
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [2,3] [2,3] [0,1] [#,#] [#,#]	:	3
[0,0] [0,0] [0,0] [5,5] [5,5] [#,#] [#,#] [5,5] [4,5] [0,1] [0,1] [#,#] [#,#]	:	3
...	:	...

Rule Matching

Figure 1: Simple rule matching example

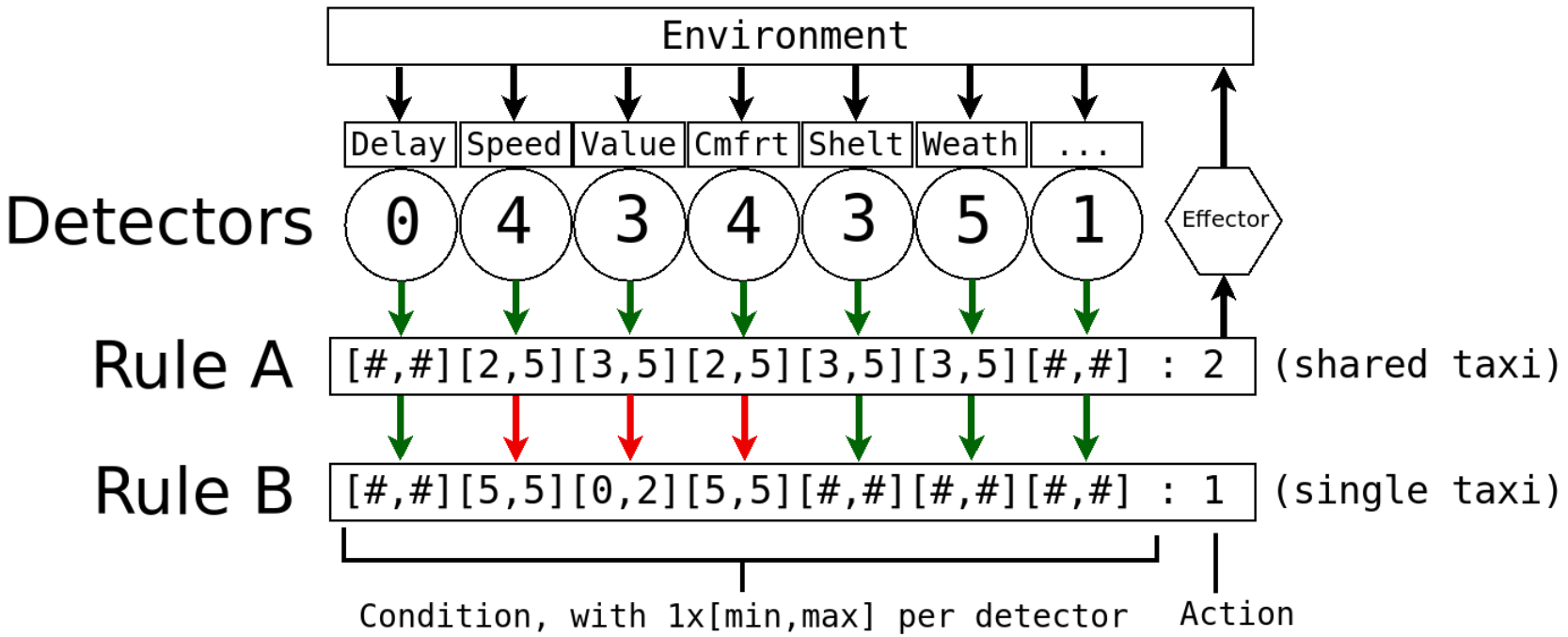
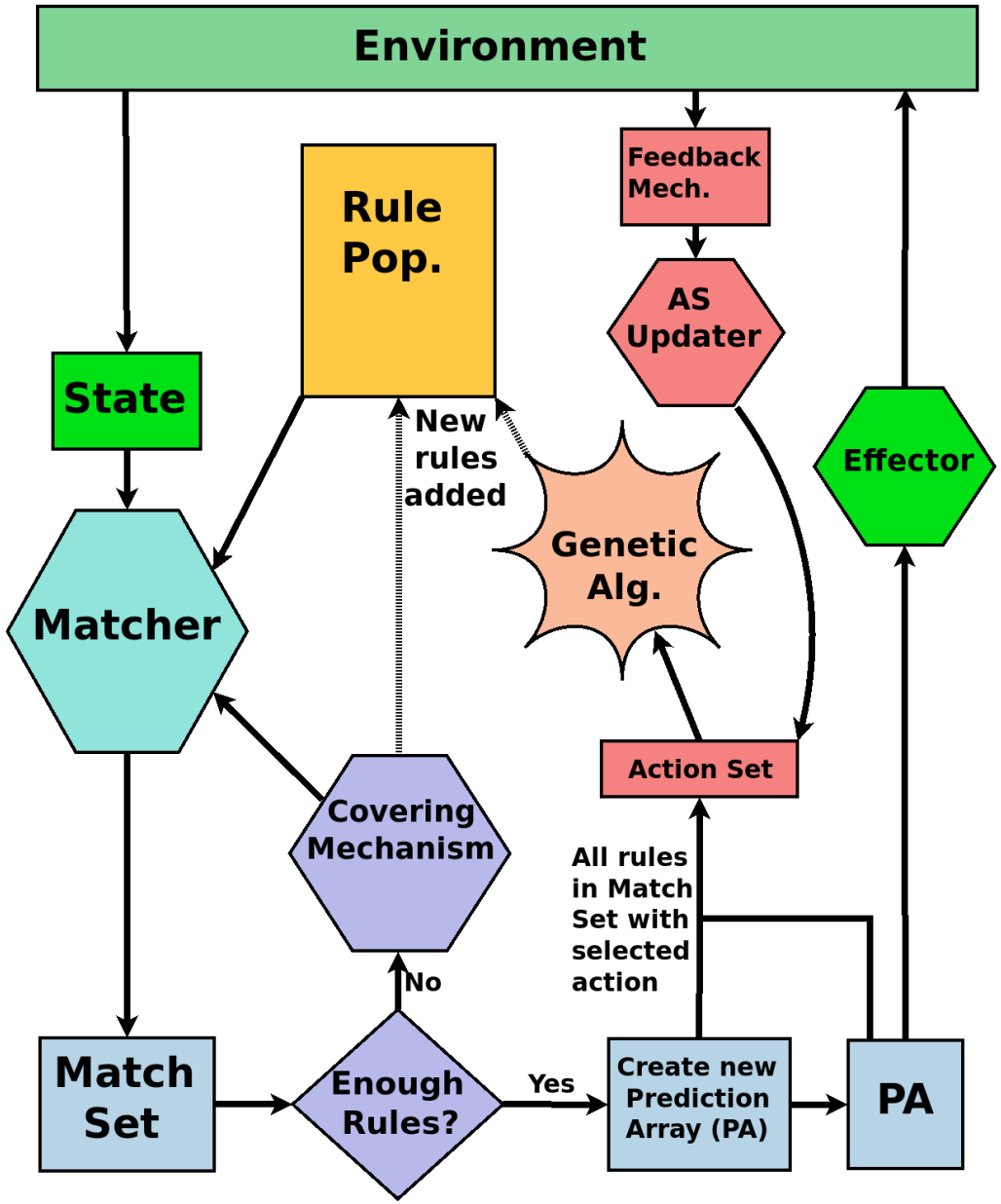


Figure 2: XCSI overview



Effector and Feedback Mechanism

- Effector outputs:
  - no change (**0**), single taxi (**1**), shared taxi (**2**), bus (**3**), boat (**4**), tube (**5**), train (**6**)
- Feedback Mechanism, receives:
  - 1000 for a correct suggestion
  - 0 for an incorrect suggestion
- This feedback is used to update the action set rules

XCS details in: Butz, Martin V., and Stewart W. Wilson. "An algorithmic description of XCS." International Workshop on Learning Classifier Systems. Springer, Berlin, Heidelberg, 2000.

Experiments

- Simulation based on London tube network
- 300 artificial ‘passengers’ with randomised:
  - preferences [0 to 5]
  - origin location / station
  - destination location / station
- Each passenger takes multiple journeys

Simulation details (1)

- Random starting time step (0 to 99) for each passenger
- Each time step has weather [0 to 5; random]
- Train, boat, bus and taxi [0 or 5; random]
- Passenger is at one node each time step
- In each time step 5% of links are out-of-action
- Interleaved... (see next 3 slides)

Simulation details (2)

Figure 3: Input list production, step 1 – order

Time Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Journey 1	S	S	S	S	S	S											
Journey 2		S	S	S	S	S	S	S	S								
Journey 3					S	S	S	S	S	S	S	S	S	S	S	S	
Journey 4	S	S	S	S	S	S	S	S									
Journey 5			S	S	S	S	S										
Journey 6							S	S	S	S	S	S	S	S	S		
Journey 7	S	S	S	S	S												

S = 'journey state'

Simulation details (3)

Figure 4: Input list production, step 2 – shuffle

Time Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Journey 3					S	S	S	S	S	S	S	S	S	S	S	S	
Journey 1	S	S	S	S	S	S											
Journey 7	S	S	S	S	S												
Journey 6							S	S	S	S	S	S	S	S	S		
Journey 4	S	S	S	S	S	S	S	S									
Journey 2		S	S	S	S	S	S	S	S								
Journey 5			S	S	S	S	S										

S = 'journey state'

Simulation details (4)

Figure 5: Input list production, step 3 – obtain input vectors

Time Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Journey 3					S	S	S	S	S	S	S	S	S	S	S	S	
Journey 1	I1	I4	I8	S	S	S											
Journey 7	I2	I5	I9	S	S												
Journey 6							S	S	S	S	S	S	S	S	S		
Journey 4	I3	I6	...	S	S	S	S	S									
Journey 2		I7	S	S	S	S	S	S	S								
Journey 5			S	S	S	S	S										

S = 'journey state'

IX = 'Input X'

Simulation details (5)

Input is checked against the 'real world' preferences of the simulated customers to get a correct input output pair...

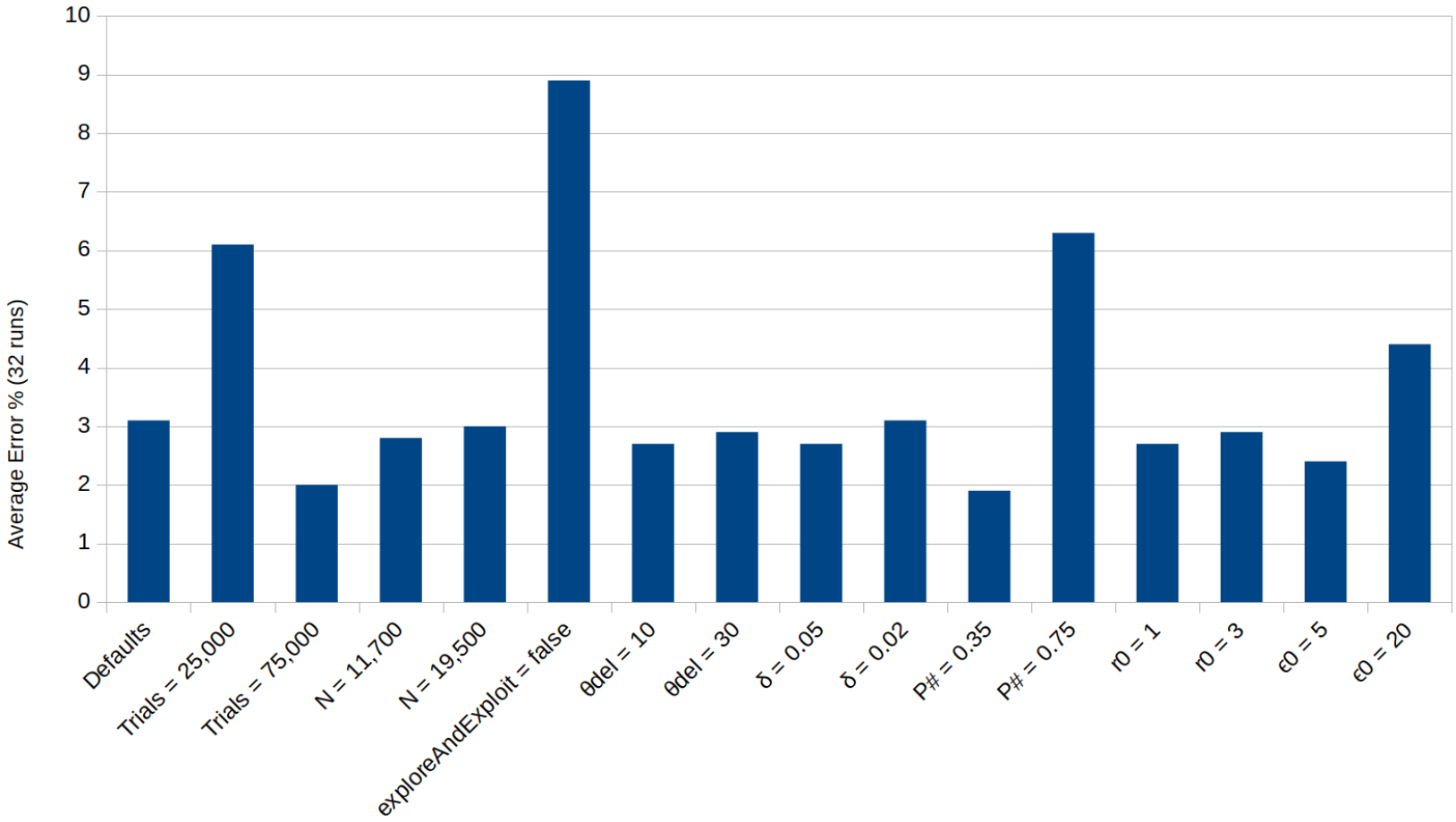
Condition	:	Action
(0)(0)(0)(5)(5)(3)(2)(5)(1)(0)(3)(4)(4)	:	?
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][2,3][4,5][#,#][#,#][#,#]	:	1
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][0,1][0,1][2,3][#,#][#,#]	:	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][4,5][0,1][0,1][#,#][#,#]	:	3
...	:	...

Correct input–output pair in this example: 000553251034 4 : 2

We therefore assemble a list of inputs and answers ($>$ 51,000) for training and testing XCSI.

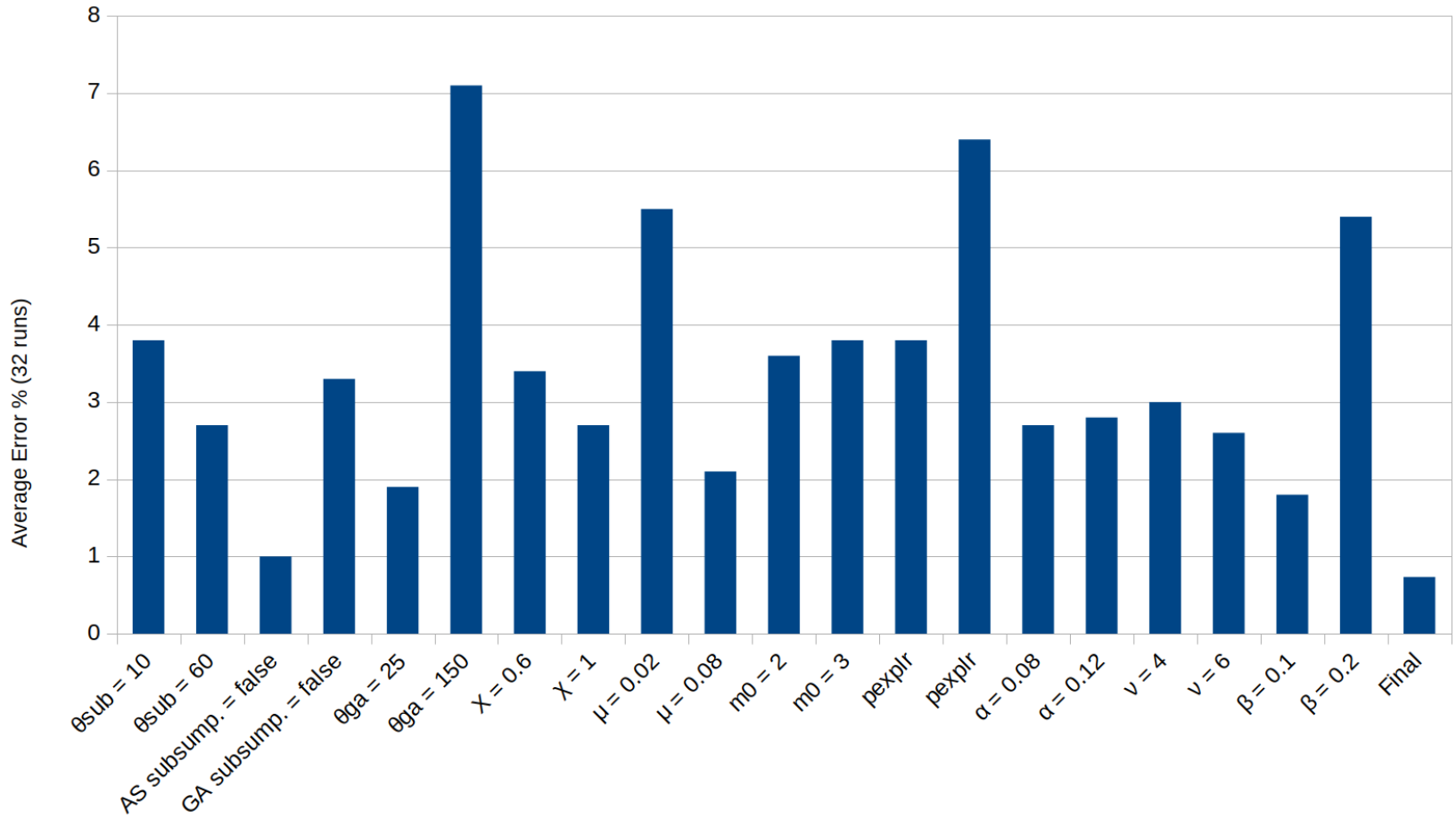
Results (1)

Figure 6: Error % for different parameter settings (1 of 2)



Results (2)

Figure 7: Error % for different parameter settings (2 of 2)



Final Parameters (Adjusted Only)

Parameter	Value	Brief Description
N	11700	Rule population size
$P_{\#}$	0.35	Probability of hash
ϵ_0	5	Error threshold
θ_{ga}	25	Genetic algorithm frequency
θ_{del}	10	Deletion threshold
β	0.1	Affects update of p,ϵ, and action set size for classifiers
α	0.08	Affects fitness updates
ν	6	Affects fitness updates
χ	1	Likelihood of GA crossover operation
μ	0.08	Likelihood of GA mutation operation
δ	0.05	Modifies the effect of fitness on classifier 'deletion vote'
θ_{sub}	60	Subsumption threshold
AS subsumpt.	false	Perform subsumption in the action set?

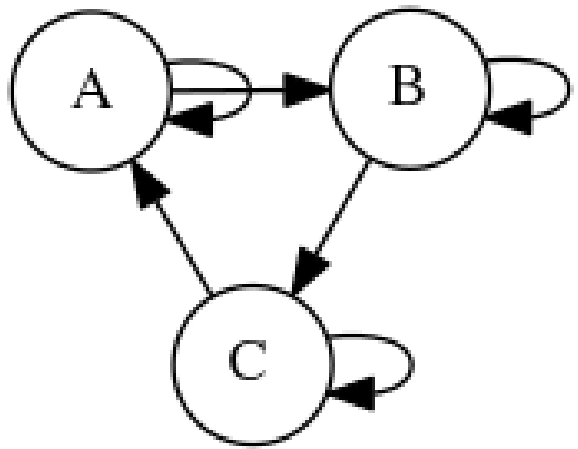
Final Performance

- Minimum trial error: 0.1%
- Average error: 0.734%
- Maximum trial error: 2%
- In concrete terms, over 99% of passengers would get the correct suggestion.

Outline

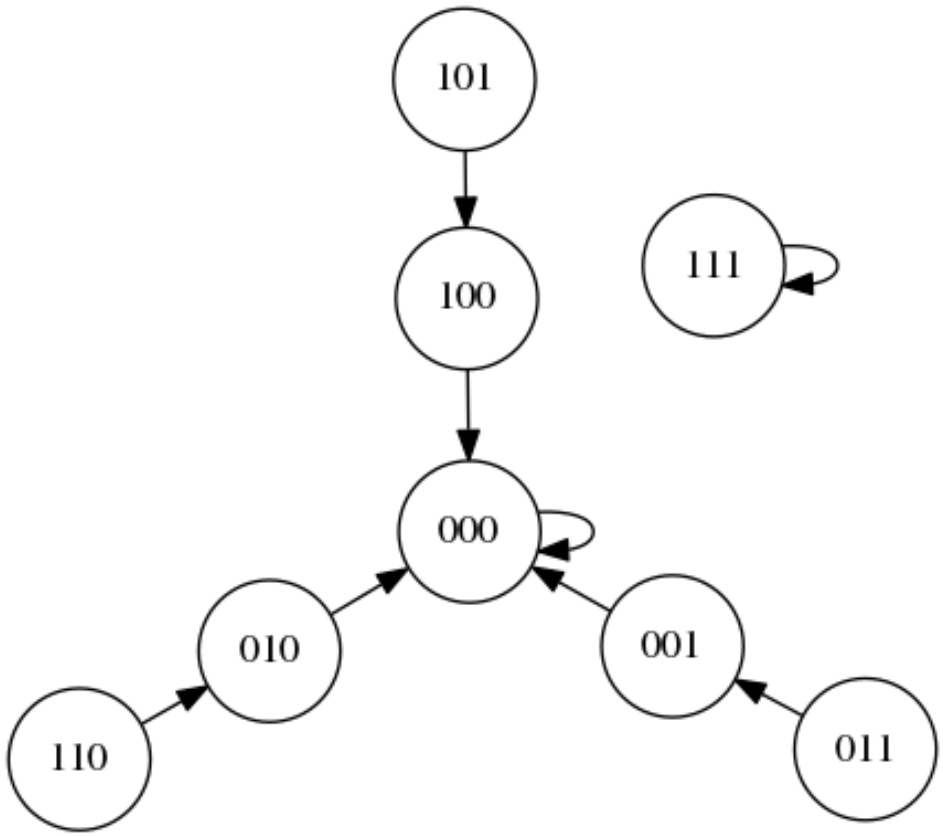
Random Boolean Networks (RBNs)

Figure 8: A Random Boolean Network (RBN) with N=3, K=2



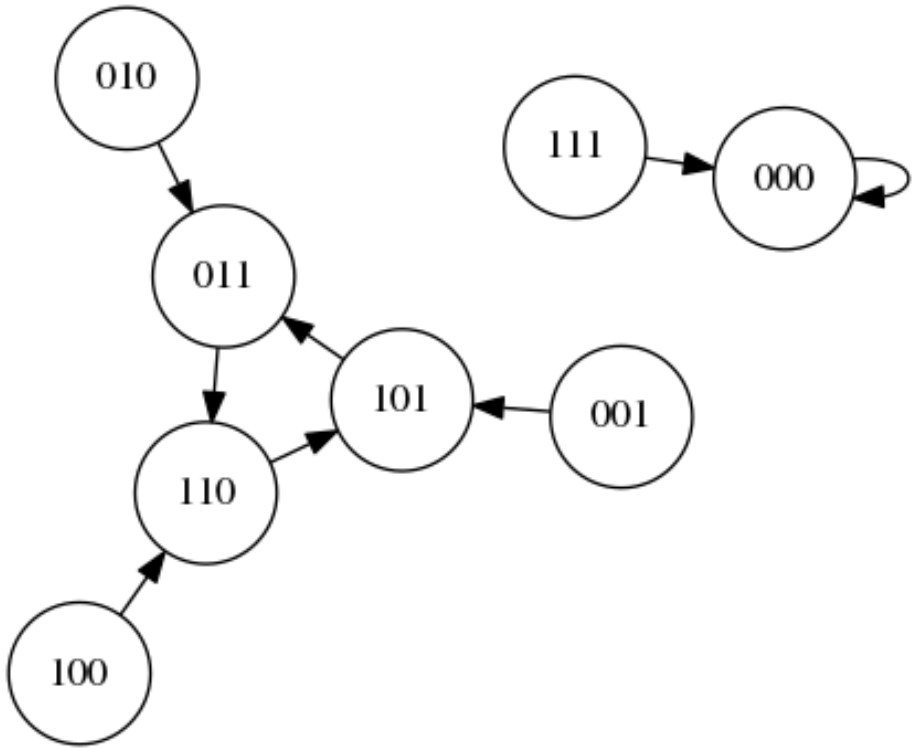
Controllability in RBNs (1/3)

Figure 9: State space of RBN of Fig. 8 (all AND); two attractors



Controllability in RBNs (2/3)

Figure 10: State space of RBN of Fig. 8 (all XOR); two atrractors



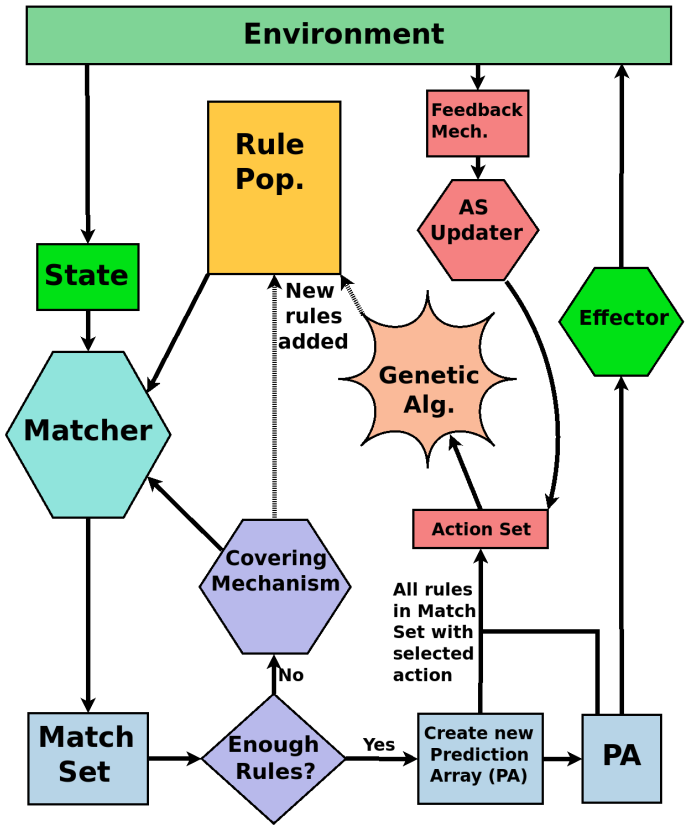
Controllability in RBNs (3/3)

- The general notion of *control* in complex networks:
  - The ability to direct the network from any state to a target state.
- can be refined to:
  - the ability to direct an RBN from any state to (one of) its attractors

The objective is to evolve a rule set that directs an RBN from any state to (one of) its attractors.

XCS – overview

Figure 11: XCS - condition is a bit string



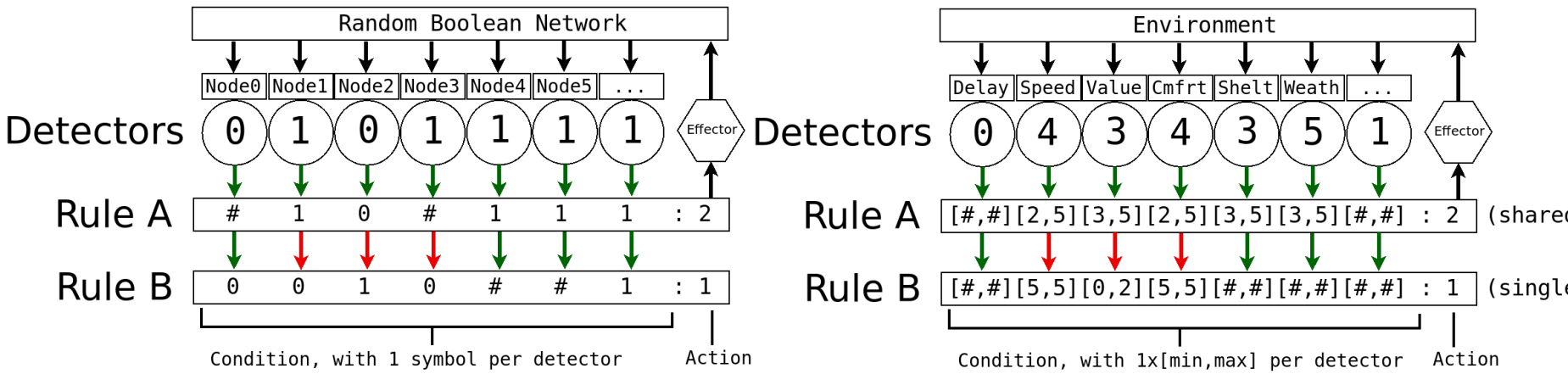
XCS details in: Martin V. Butz and Stewart W. Wilson. "An algorithmic description of XCS." Int'l Workshop on *Learning Classifier Systems*. Springer, Berlin, 2000.

Applying XCS to control RBNs

- Each rule represents a *condition : action* expression that links specific states of the RBN (conditions) to bit flips (actions)
- To shift from single state to the state cycle attractor, apply one of `###:1; ###:2; ###:3`
- To shift from the state cycle to the single state attractor, apply one of `110:3; 011:1; 101:2; 001:3; 010:2; 100:1`
  - where # denotes "don't care" and the action represents the index of the bit to flip

XCS, not XCSI

Figure 12: Rules in XCS



Controlling RBNs using XCS

Figure 13: Control graph for a N=5,K=2 RBN using XCS

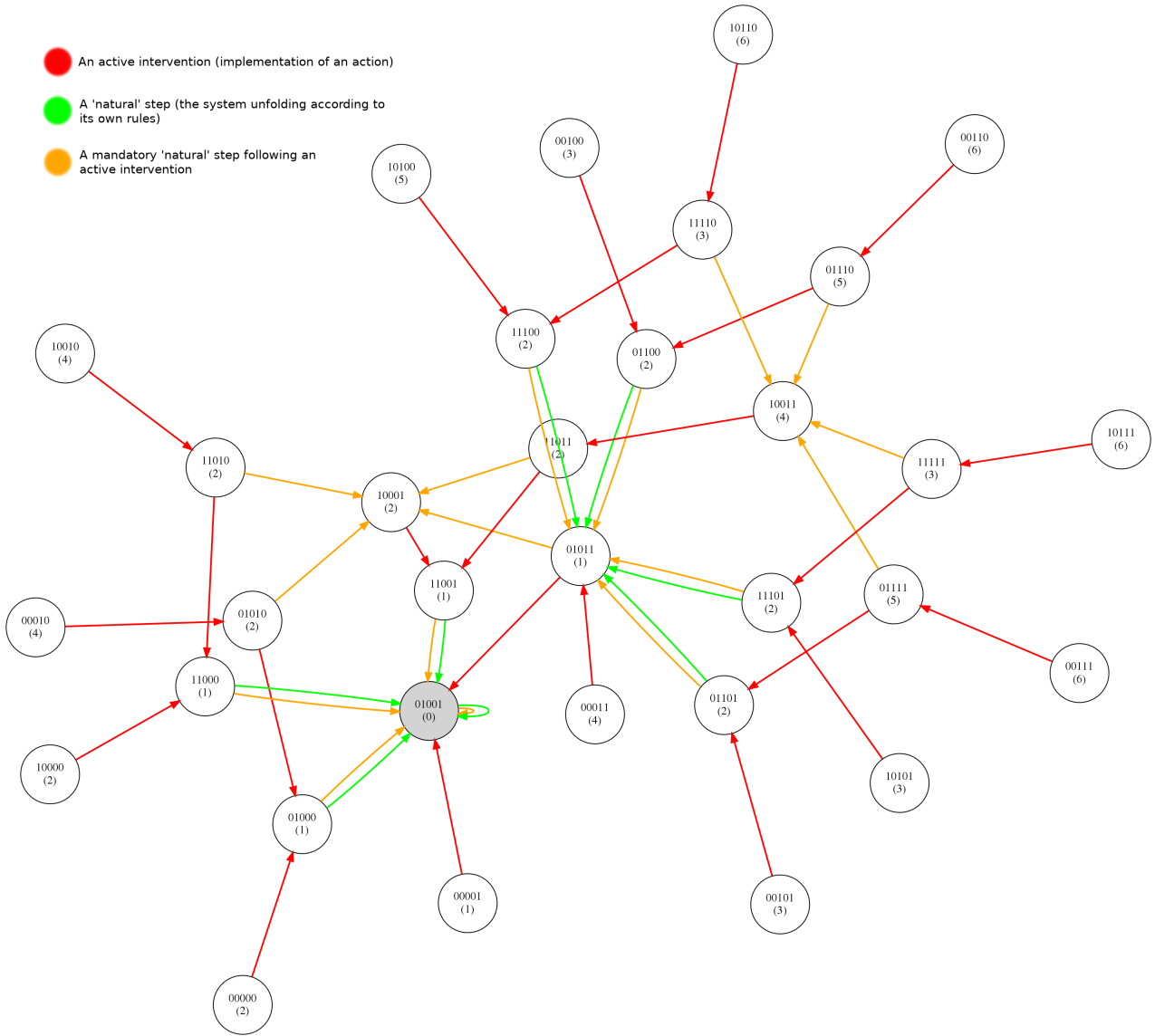


Figure for OCR vs XCS

*Wilson, Stewart W. "Mining oblique data with XCS." International Workshop on Learning Classifier Systems. Springer, Berlin, Heidelberg, 2000.

Outline

System Dynamics Models

- System dynamics models[1] are often understood as complex networks
- *Control* involves interventions on the values of state variables (actions) to steer the network to a desired state
- XCSR (XCS *Real*): real valued extension of XCS

Question: Can we use XCSR to identify correct actions on *control* nodes[2] for steering a dynamical system to a desired state ?

---

[1] J.D. Sterman "Business Dynamics: Systems Thinking and Modelling for a Complex World", McGraw-Hill, New York, 2000

[2] S. Moschoyiannis, N. Elia, A. Penn, *et al* "A Web-based Tool for Identifying Strategic Intervention Points in Complex Systems ", ETAPS 2016

Population growth model (1/2)

The S-shaped population growth model[3]:

$$\frac{\text{dPopulation}}{\text{dt}} = \text{Birth Rate} - \text{Death Rate} \tag{1}$$

$$\text{Birth Rate} = \text{FractionalBR} * \text{Population} \tag{2}$$

$$\text{Death Rate} = \text{FractionalDR} * \text{Population} \tag{3}$$

$$\text{FractionalBR} = 1 - \frac{1}{1+e^{-7*(\text{PCC}-1)}} \tag{4}$$

$$\text{PCC} = \frac{\text{Population}}{\text{Carrying Capacity}} \tag{5}$$

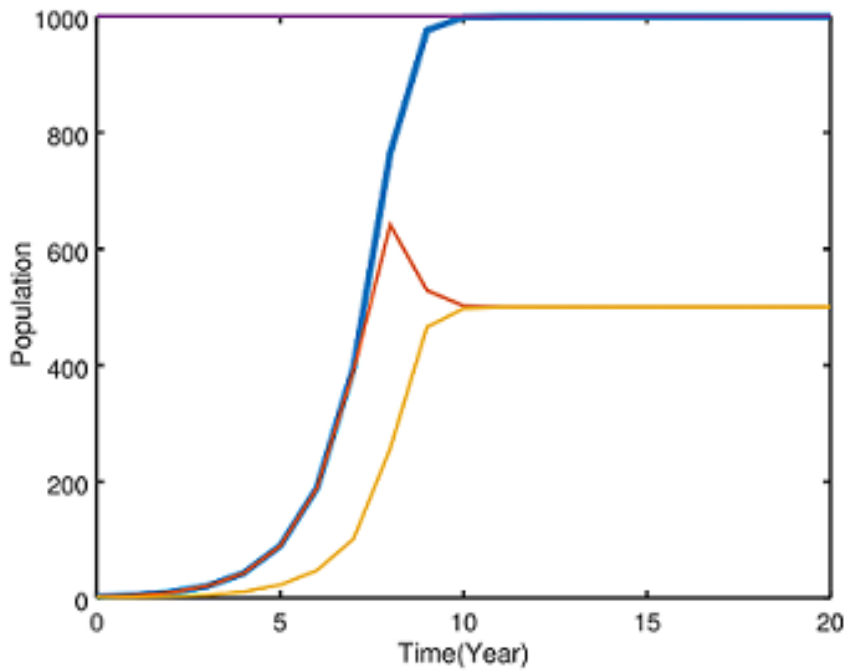
and the value of Carrying Capacity is a constant.

[3] J.D. Sterman "Business Dynamics: Systems Thinking and Modelling for a Complex World", McGraw-Hill, New York, 2000

Population growth model (2/2)

- System has only one state variable, thus one attractor where the rates converge

Figure 14: Population (blue) converges at the value of Carrying Capacity (grey); BirthRate(red) and DeathRate(yellow) converge at half that value



Controlling this model (1/2)

- Establish *continuous* control
  - *Step* defined as the change of value in state variable after one unit of time
- XCSR selects an action before each step
  - Action set: *increase/decrease* Birth- and Death-Rate by a constant or percentage, or no action for a given step
- Set *explore / exploit* to allow for 50% chance of selecting random action

Controlling this model (2/2)

XCSR can bring the system to a desired stable state (attractor)

- **But** the evolved rule set contains *overgeneralised* rules (too many #'s)
- **Revisited** reward mechanism – generalisation from mutation only
- **But** when current state is "too far" from desired state => *action chain learning problem*
- **Revisited** notion of *step*: action selection followed by full simulation until convergence

Lotka-Volterra predator–prey model (1/2)

This model[4] is a well known non-linear first order differential equations couple:

$$\frac{\text{dPrey}}{\text{dt}} = \text{Prey} * (\text{PreyBirthRate} - \text{DeathRateperPredator} * \text{Predator}) \quad (6)$$

$$\frac{\text{dPredator}}{\text{dt}} = -\text{Predator}(\text{PredatorDeathRate} - \text{BirthRateperPrey} * \text{Prey}) \quad (7)$$

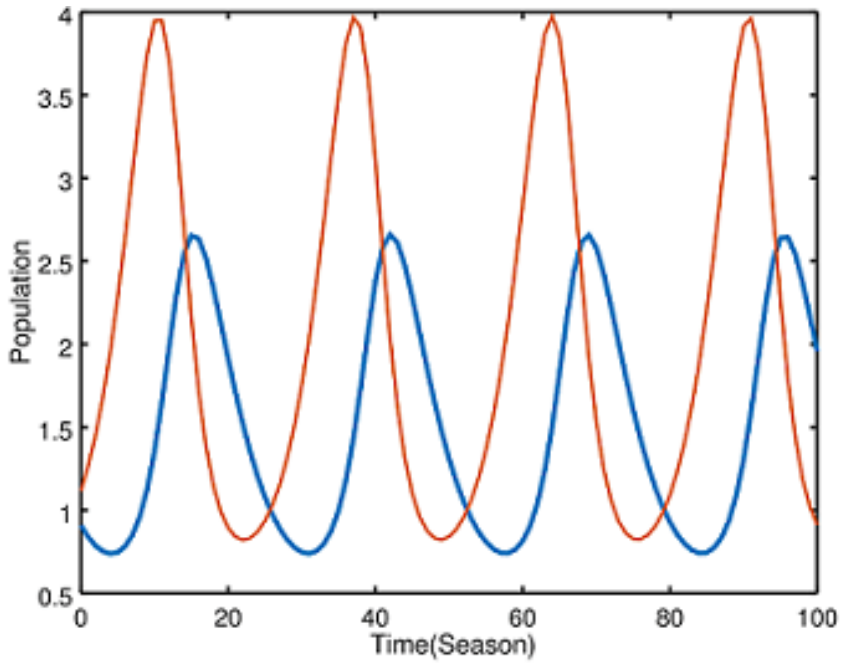
where PreyBirthRate, DeathRatePerPredator, PredatorDeathRate and BirthRatePerPrey are constants.

[4] A.J. Lotka "Elements of Mathematical Biology" Dover Publications, New York, 1956

Lotka-Volterra predator–prey model (2/2)

- System has two state variables, thus more than one attractors

Figure 15: Two state variables imply multiple attactors; Predator(red) and Prey(blue) populations have different attractors



Controlling the L-V model (1/2)

XCSR generates logically correct classifiers, and can bring the system to a desired stable state (attractor)

- **But** when more than 10 consecutive good actions are needed
  - XCSR not able to reach a reward in the early runs
  - overgeneralised rules with prediction of 0 reward
  - hence, these rules are highly accurate, hence their fitness inflated
  - leads to *action chain learning problem*
- **Revisited** notion of action – using adaptive step sizes seems promising; *adaptive action mapping (XCSAM)*

Conclusions

- XCSR can control system dynamics models, under certain conditions
  - challenge: **multi-step** problems, **continuous** control
- To address the *action chain learning problem*, convert the multi-step problem to multiple single step problems
- **However**, not feasible for higher order systems
  - Try adaptive step sizes => XCSAM
  - Try discretising the system state
- **Rule-based machine learning** (RBML) is human readable
  - why XCS* suggested a given action
  - how XCS* arrived at suggesting that action

Thanks/Acknowledgements

- Thank you for your attention
- Thanks to Alastair Finlinson, Sophia Manalo, George Papagiannis
- This research was partly funded by
  - the Department for Transport, via Innovate UK and the *Accelerating Innovation in Rail* (AIR) Round 4 programme
  - the UKRI EPSRC council, and the AGElink project
  - the NCSC, and two research summer internsnips
  - the EIT Digital, and the Real-Time Flow project