TUTORIAL

Learning Classifier Systems

Martin V. Butz

Department of Cognitive Psychology III
University of Würzburg
Röntgenring 11, 97070 Würzburg, Germany

butz@psychologie.uni-wuerzburg.de
http://www.psychologie.uni-wuerzburg.de/i3pages/butz/

Genetic and Evolutionary Computation Conference GECCO 2007

07/07/2007

Overview – Part 1

PART1: General Introduction

- **Historical remarks**
- **LCSs: Framework and basic components**
  - Problem types
  - Michigan- and Pittsburgh-style LCSs
  - Knowledge representations
  - Learning in LCSs
  - Questions to consider.

Part 2: LCS Systems and Concepts

- The XCS classifier system
- Anticipatory learning classifier systems
- Other learning classifier systems
- Summary, conclusions, & further information

Historical Remarks

- **Proposed and introduced by John H. Holland**
  - In the 1970s
  - Schema processing mechanism (Holland, 1975)
  - Cognitive systems (CS1, Holland, & Reitman, 1978)
- **First applications in the 1980s**
  - Poker decisions (Steve F. Smith, 1980)
  - Animal-like automaton (Booker, 1982)
  - Gas pipeline control task (Goldberg, 1983)
  - Video-eye focusing (Wilson, 1983)
  - Animat automation (Wilson, 1985, 1987)
  - Others (cf. Goldberg, 1989)

LCS Renaissance Since 1990s

- **Introduction of two fundamental Michigan-style LCS systems:**
  - The strength-based ZCS system (Wilson, 1994)
  - The accuracy-based XCS system (Wilson, 1995)
- **Since late 1990s:**
  - New LCS representations
  - New RL-based and gradient-based prediction formation
  - Advanced understanding of genetic algorithms
  - Comparisons with other machine learning techniques
  - Competitive LCS results in benchmark classification, function approximation, and reinforcement learning problems

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
ACM 978-1-59593-698-1/07/0007.

LCSs: Frameworks and Basic Components

1. Problem types
2. Michigan and Pittsburg-style LCSs
3. Knowledge representation
4. Learning in LCSs
5. How an LCS works
6. Questions to consider

Problem Types

1. Classification problems
2. Reinforcement learning problems
3. Function approximation problems
4. General prediction problems

Problem Types: Classification Problems

- Task:
  Find a *compact* set of rules that classify all problem instances *maximally accurately*.
- Examples:
  - Medical diagnosis
  - Image classification
  - Game analysis
  - Mushroom classification
  - Boolean functions

Rules for mushroom classification:

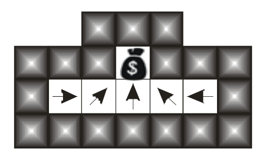
Condition	Classification
Small AND green	edible
Small AND pink	poisonous
Large AND green	poisonous
Red AND Has-spots	poisonous
…	

Problem Types: Reinforcement Learning Problems

(cf. Sutton, & Barto, 1998)

- Task:
  Find an *optimal behavioral policy* represented by a *compact* set of rules.
- Examples:
  - Maze tasks:
    Find the food, avoid predators
  - Mountain car problem:
    Drive the car to the top of the hill
  - Blocks world problems:
    Move the blocks to a goal constellation
  - *POMDPs pose additional challenges.*

Solution for a simple maze task

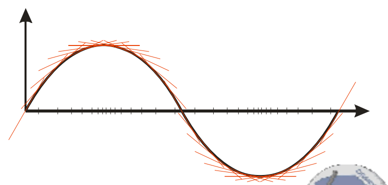


Problem Types: Function Approximation Problems

- Task:
  Find an *accurate function approximation* represented by a partially overlapping set of approximation rules.

- Examples:
  - Constant approximation of a step function
  - Piece-wise linear approximation of a sine function

Piece-wise linear solution for a sine function



Problem Types: Solving Any Prediction Problem

- LCSs can generally solve any type of prediction problem.
  - Conditions cluster the problem space.
  - Predictions form inside the evolving clusters.
- Feedback can be either immediate or delayed.
  - Given delayed feedback, feedback propagation is necessary.

Michigan- and Pittsburgh-style LCSs

1. Fundamental system differences
2. Targeted problem solutions

Pittsburgh- vs. Michigan-style LCSs: Fundamental Differences

Michigan-style LCS	Pittsburgh-style LCS
One complete problem solution is encoded.	Each individual encodes an entire problem solution.
Each individual encodes one single rule.	Each individual encodes an entire set of rules.
Rules are evaluated (competitively) individually.	Whole rule sets are evaluated.
Rules evolve (competitively) individually.	Complete competing problem solutions evolve.
An online learning system that learns iteratively from single problem instances.	An offline learning system that learns iteratively from sets of problem instances.
Typically, solutions with a larger number of (local) rules evolve.	Typically, small rule sets evolve.

Michigan vs. Pittsburgh-style LCSs
Targeted Problem Solutions

Michigan-style LCS

- Fundamental properties
  - Evaluates rules locally.
  - Optimizes rules locally.
- Major qualities
  - Distributed, locally optimal problem solution
  - Combines local gradient-based approximation with local evolutionary rule-structure optimization.

Pittsburgh-style LCS

- Fundamental properties
  - Evaluates and optimizes rule-sets globally (based on sets of problem instances).
- Major qualities
  - Evolves one global problem solution.
  - Mainly uses evolutionary rule structure optimization.
- Arguable actually a GA rather than an LCS.

Knowledge Representation

1. Population-based knowledge representation
2. Condition structures
3. Prediction structures
4. Examples

Population-Based Knowledge Representation

- Population (set) of classifiers (rules)
  - Usually unordered
- Classifiers with
  - Condition part **C**
  - (Action part **A**)
  - Prediction part **P**
  - Meaning:
    *"If condition **C** is satisfied (and action **A** is executed), then **P** is expected to be true."*
- Given a problem instance
  - Solution is determined by *matching* classifiers (those whose conditions are satisfied).

Condition Structures I

(conditions also called "taxa")

- For binary problems
  - Ternary alphabet 0, 1, #
  - Examples:
    - (100#) *matches* 1000 and 1001
    - (#1#) *matches* 010, 011, 110, 111
- For real-valued problems
  - Interval encoding
  - Hyperellipsoidal encoding
  - Example (interval encoding):
    - ([0,.5][.2,.7][0,1]) *matches if* att.1 has a value between 0 and .5, att.2 between .2 and .7, and att.3 between 0 and 1.

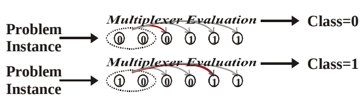
Condition Structures II

- Nominal problems
  - Set-based encoding
  - Interval encoding
  - Example (set-based encoding):
    - ({a,b,d},{b}) *matches if* att.1 equals 'a', 'b', or 'c' and att.2 equals 'b'
- Mixed-valued problems
  - Mixed encodings
- Other condition representations
  - Partial matching (Booker, 1985)
  - Default hierarchies (Holland et al., 1986)
  - Fuzzy conditions (Bonarini, 2000; Valenzuela-Rendón, 1991)
  - Neural-network-based encodings (Bull, O'Hara, 2002)
  - GP tree encodings with S-expressions (Lanzi, 1999)

Prediction Structures

- Traditionally, a constant value prediction
  - Given conditions are satisfied, value P is predicted.
- For real-valued function approximation problems
  - Linear predictions (weight vector with offset)
  - Polynomial predictions
- Generally
  - Predictions can be computed based on available problem input.
  - Predictions are usually learned by means of gradient-based learning techniques (problem of "credit assignment")

Solution Representation Examples: Multiplexer Problem

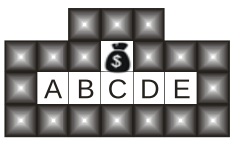


Problem instance	Class
000000	0
001000	1
000111	0
011011	0
101101	0
100010	1
100101	0
110000	0
…	…

Optimal solution representation

C	A	P
000###	0	1000
001###	1	1000
01#0##	0	1000
01#1##	1	1000
10##0#	0	1000
10##1#	1	1000
11###0	0	1000
11###1	1	1000

Solution Representation Examples: Simple Maze Problem



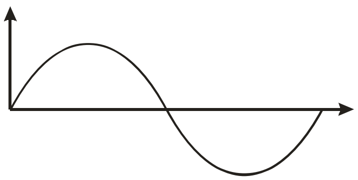
Problem instances sampled running through the maze

State	Sensation	↑	↗	→	↘	↓	↙	←	↖
A	11011111	A	A	B	A	A	A	A	A
B	10011101	B	F	C	B	B	B	A	B
C	01011101	F	C	D	C	C	C	B	C
D	11011100	D	D	E	D	D	D	C	F
E	11111101	E	E	E	E	E	E	D	E

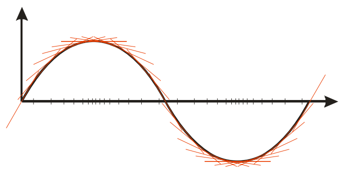
Optimal solution representation (with reward propagation)

C	Matches	A	P
11#####1	A,E	↑	810
1#0###0#	B,D	↑	900
0#######	C	↑	1000
11#####1	A,E	↗	810
#10###0#	C,D	↗	900
#0######	B	↗	1000
##0####1	A,B,C	→	900
11####0#	D,E	→	810
11#####1	A,E	↘	810
…		…	…

Solution Representation Examples:
Function Approximation Problem



Problem instances sampled from
f(x) = sin(2πx) with x in [0,1]



Good solution representation

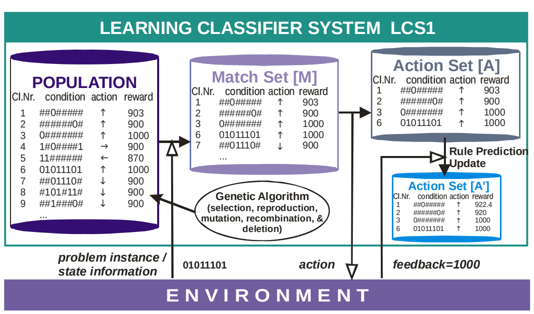
C	P
[.00 , .08]	.00 , 6.3
[.05 , .14]	.33 , 5.2
[.12 , .19]	.68 , 3.6
[.18 , .22]	.90 , 1.9
[.21 , .24]	.96 , 0.9
[.24 , .26]	.98 , 0.0
[.26 , .29]	.98 , -0.9
[.28 , .32]	.97 , -1.9
[.31 , .38]	.92 , -3.6
[.36 , .45]	.72 , -5.2
[.42 , .58]	.50 , -6.3
...	

Learning in LCSs

1. Basic operation cycle
2. Prediction estimation
3. Rule quality evaluation
4. Rule structure evolution
5. Interplay of rule evaluation and rule learning

Basic Operation Cycle In LCSs

- Repeat until done
  - Get current problem instance (input) & form match set
  - Decide on classification / action, execute action, form action set
  - Receive feedback & update rule estimates
  - Apply GA

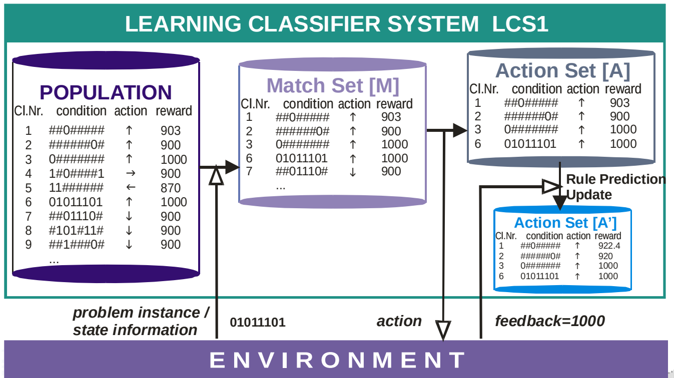


Prediction Estimation

(also called "credit assignment subsystem")

- Gradient-based prediction updates
- For constant predictions:
  - Original method: Bucket Brigade algorithm (Holland, 1985)
  - "Modern" techniques:
    - Q-learning derived updates
    - Widrow-Hoff rule (Widrow, Hoff, 1960)
  - Generally, an iterative prediction update based on prediction error
- For linear predictions:
  - Delta rule
  - Better: Recursive least squares or Kalman filtering
- For other prediction types:
  - Use best local (gradient-based) approximation technique

Example
Simple Prediction Update



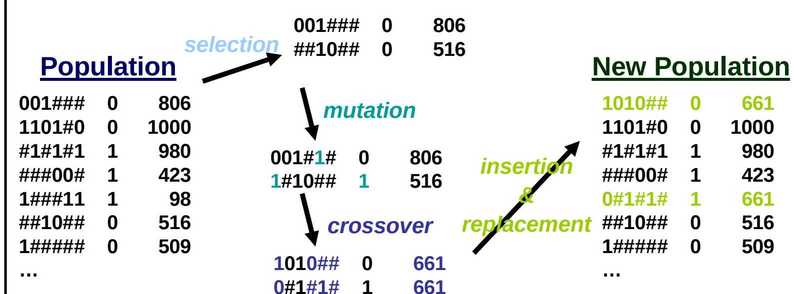
Rule Quality (Fitness) Estimation

- Rule quality is derived from rule prediction.
- Iterative rule quality update
- Originally:
  - Rule quality = rule prediction (strength-based update, problem of *strong overgenerals*, Kovacs, 2004)
- Now often:
  - Rule quality = average (shared) payoff received (*shared, strength-based*) (see: ZCS system, Wilson, 1994)
  - Rule quality = accuracy of prediction (accuracy-based) (XCS system, Wilson, 1995)

Rule Structure Evolution

- Rule structure evolves by means of a genetic algorithm (GA) (possibly plus heuristics).
  - Usually **constant** population size
  - Fitness = rule quality
  - Steady state GA: selection of few highly fit classifiers
    - Different selection methods possible
    - Often niche-based selection
  - Mutation, crossover applied to rule condition (and action)
  - Insertion of offspring
  - Deletion of low-fitness classifiers

Example:
Iterative Rule Structure Evolution



Rule Quality Estimation and Rule Structure Evolution

- Gradient-based rule quality estimation
  - **Goal**: Fast identification of current best classifiers
    - Fast and maximally accurate parameter estimates
    - Fast adaptation to population and environment dynamics
- Evolutionary rule structuring (possibly combined with heuristics)
  - **Goal**: Effective search through promising solution structure subspaces
    - Effective selection
    - Effective local neighborhood search
    - Effective substructure propagation and recombination

How Does an LCS Work? Interplay of Estimation and Evolution

- Successful rule structure evolution depends on effective rule quality estimation (fitness).
- Thus, optimal problem solution structure can only evolve effectively if:
  - Rule quality is determined as fast as possible.
  - Thereby, mind the explore-exploit dilemma (need to evaluate all rules)!

Questions to Consider

1. Which LCS should I use?
2. How can I optimize my LCS?

Which LCS should I use?

- Consider the problem solution representation
  - *Can local approximations yield an effective global solution to the problem at hand?*
    Yes: Michigan-style LCSs will be effective.
    No: Consider also using Pittsburgh-style LCSs, GP, or other related optimization techniques.
- Consider the problem type
  - *Do you want to learn iteratively online or offline?*
    - Online: Another reason to use Michigan-style LCSs. (also others possible, though)
    - Offline: Both LCS systems can be applied.

How can I optimize my LCS?

- Given a problem and a targeted solution representation:
  - How should I partition the problem space?
    - What is the best condition representation?
    - How can I evolve condition structures maximally effectively?
  - What do I want to predict?
    - What is the best prediction representation?
    - How do I approximate predictions and derive fitness most effectively?
  - How is feedback available?
    - Is feedback available immediately (one-step problems)?
    - Is feedback delayed but fully predictable (MDP)?
    - Is feedback delayed and only partially predictable (POMDP)?

… these questions will now be addressed in concrete LCS implementations.

Any other questions so far?

Overview – Part 2

PART1: General Introduction
- Historical remarks
- LCSs: Framework and basic components

Part 2: LCS Systems and Concepts
1. **The XCS classifier system**
   - Framework & functionality
   - XCS – Performance Suite
2. **Anticipatory learning classifier systems**
   - Introduction
   - ACS2
   - XACS
   - Potentials
3. **Other classifier systems**
4. **Summary, conclusions, & further information**

The XCS Classifier System

- Introduced by Stewart W. Wilson (1995)
- Is a Michigan-style LCS
- Major novelties:
  - Q-learning based reinforcement learning
  - Relative accuracy-based fitness
  - Action-set restricted selection (niche selection)
  - Panmictic (population-wide) deletion

XCS: Framework & Functionality

1. Framework overview
2. Evolutionary pressures
3. Solution representation
4. Problem bounds
5. Condensation and Compaction
6. Summary

Classifiers

- Condition Part **C**
  → *When classifier is applicable*
- Action Part **A**
  → *Which action to execute*
- Prediction **P**
  → *Expected average reward*
- **Prediction Error ε**
  → *Estimate of mean absolute deviation of P*
- **Fitness F**
  → *Estimate of average action-set-relative accuracy of P*

Additional parameters:

- Action set size estimate **as**
- Time stamp of last GA application **ts**
- Experience **exp**
  → *How often parameters were updated.*
- Numerosity **num**
  → *How many identical classifiers are represented.*

Parameter Updates

situation **S**	action **A**	feedback **R(S,A,S_{+1})**
classifier **cl**	condition part **C**	reward pred. **P**
prediction error **ε**	fitness **F**	learn. rate **β**
discount factor **γ**	min. error ε_0	accuracy modifiers **α, η**

Prediction array determination

$$P(S,A) = R(S,A,S_{+1}) + \gamma \max_{A_{+1}} \frac{\sum_{cl \in [A(S_{+1},A_{+1})]} cl.F \cdot cl.P}{\sum_{cl \in [A(S_{+1},A_{+1})]} cl.F}$$

Prediction update

$$cl.P \leftarrow cl.P + \beta(P(S,A) - cl.P)\left[\frac{cl.F}{\sum_{c \in [A(S,A)]} c.F}\right]$$

Error update

$$cl.\varepsilon \leftarrow cl.\varepsilon + \beta[|cl.P - P(S,A)| - cl.\varepsilon]$$

Current accuracy derivation

$$cl.\kappa = \begin{cases} \alpha(cl.\varepsilon / \varepsilon_0)^{-\eta} & \text{if } cl.\varepsilon > \varepsilon_0 \\ 1 & \text{otherwise} \end{cases}$$

Set-relative accuracy derivation

$$cl.\kappa' = \frac{cl.\kappa}{\sum_{c \in [A(S,A)]} c.\kappa}$$

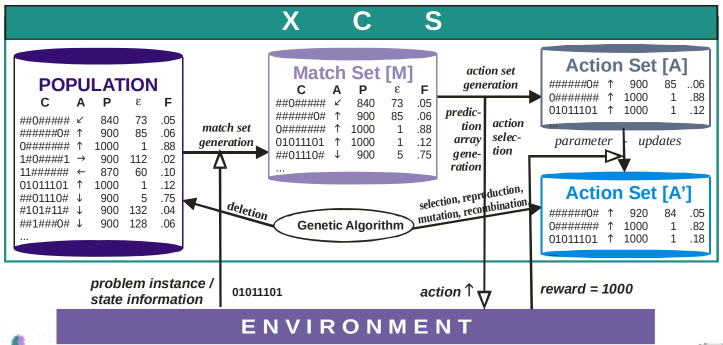
Fitness update

$$cl.F \leftarrow cl.F + \beta[cl.\kappa' - cl.F]$$

Evolutionary Algorithm

- Fixed population size
- Steady-state genetic algorithm in action sets
- Two reproductions and deletions per iteration
  - Reproduction in action set
  - Selection (proportionate or tournament) based on fitness
  - Deletion (proportionate selection) from whole population based on coverage
- Genetic operators:
  - *Mutation*
  - *Recombination*

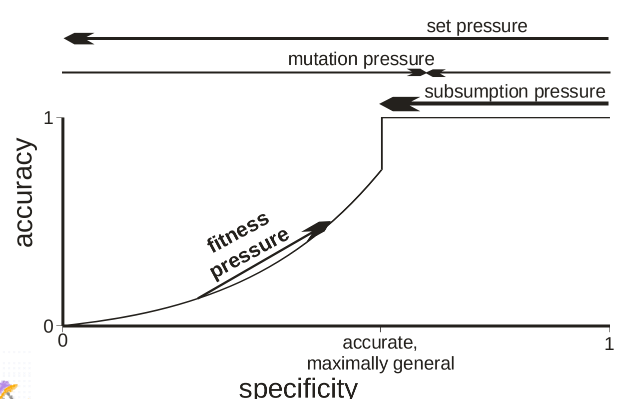
Learning Interaction



How Does It Learn?
XCS Learning Pressures

- Parameter updates identify *most accurate* classifiers.
- Genetic algorithm causes evolutionary pressures on condition structures
  - Set pressure (reproduction of more general classifiers)
  - Fitness pressure (reproduction of more accurate classifiers)
  - Mutation pressure (diversification – specificity/generality pressure)
  - Subsumption pressure (elimination of accurate, over-specialized classifiers)

Evolutionary Pressures

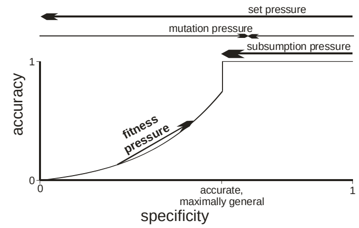


What Does it Learn?
Solution Representation

- GA propagates *most accurate* classifiers.
- Generalization pressures propagate accurate, *maximally general classifiers*.
- Niche reproduction with coverage-based deletion ensures *occurrence-based coverage*.
- Thus, XCS strives to learn a complete, maximally accurate, and maximally general approximation model.
  - In **classification problems**: Class-dependent subspace partitions.
  - In **reinforcement learning problems**: Approximation of Q-value function.
  - In **function approximation problems**: Piecewise linear function approximation.

Can We Assure Learning Success? Learning Bounds

- Proper population initialization: ***covering bound***
- Ensure supply: ***schema bound***
- Ensure growth: ***reproductive opportunity bound***
- Ensure solution sustenance: ***niche support bound***
- Enough learning time is necessary: ***learning time bound***



Ensuring Learning Bounds

- Learning bounds can be assured by
  - Setting initial specificity sufficiently low
  - Setting population size sufficiently high (problem difficulty)
  - Setting mutation properly (controlling specificity and search time)
  - Allowing enough learning iterations (time)
- PAC learning relation in k-DNF problems (Butz, Goldberg, & Lanzi, 2005)

Condensation and Compaction

- Population sizes of final solution rather large
  - GA is running continuously.
  - Many redundant and inaccurate classifiers
- Use condensation:
  - Continue to run GA without mutation and crossover (Kovacs, 1996; Wilson, 1995).
- Use closest classifier matching (CCM):
  - Avoids holes in problem coverage.
  - Matches fixed number of closest classifiers (Butz, Lanzi, & Wilson, in press).
- Greedily delete overlapping / irrelevant classifiers
  - Can be hard to determine which ones to delete.
  - Several methods are available (Butz, Lanzi, & Wilson, in press; Dixon, Corne, & Oates, 2003; Wilson, 2002).

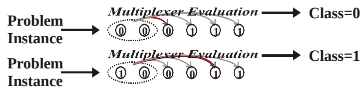
Summary of XCS Properties

- XCS represents its solution by a collection of sub-solutions (that is, a population of classifiers).
- XCS evolves a problem space clustering in its conditions.
- Clusters (subspaces) evolve to enable maximally accurate predictions.
  - Accuracy can be bounded (error threshold ε_0 and population size relation).
  - Basically any form of prediction is possible (e.g. reward, next sensory input, function value).

XCS: Performance Suite

1. Multiplexer problem
2. Datamining problems
3. Function approximation problems
4. Reinforcement learning problems
5. Summary

XCS in 6-Multiplexer Problem



Problem instance	Class
000000	0
001000	1
000111	0
011011	0
101101	0
100010	1
100101	0
110000	0
…	…

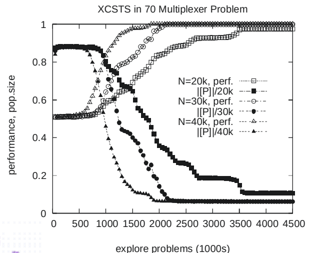
Optimal solution representation

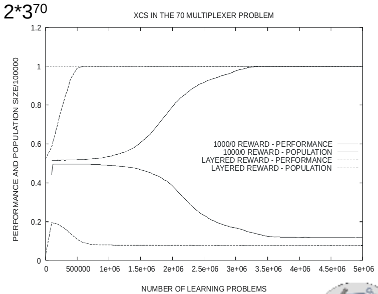
C	A	R	ε	F
000###	0	1000	0	1
000###	1	0	0	1
001###	0	0	0	1
001###	1	1000	0	1
01#0##	0	1000	0	1
01#0##	1	0	0	1
01#1##	0	0	0	1
01#1##	1	1000	0	1
10##0#	0	1000	0	1
…	…	…	…	…

Performance in MP 70

(Butz, 2006; Butz, Kovacs, Lanzi, & Wilson, 2004)

- Very hard problem
- Perfect problem solution contains 2^8=256 classifiers.
- Problem space is huge: 2^{70}
- Rule condition space is even bigger: $2*3^{70}$





Performance in Datamining Problems

(Butz, 2006)

- Conditions are encoded with attributes dependent on type of attribute in dataset (mixed encoding).
- Experiments in 42 datasets (from UCI and other sources)
- Comparisons with ten other ML systems (pairwise t-test)
- XCS learns competitively, but it is a much more general learning system.

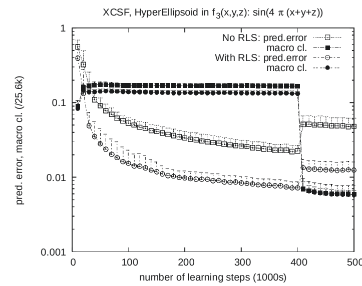
XCS	Maj.	1-R	C4.5	Naïve Bayes	PART	IB1	IB3	SMO (poly)	SMO (pol.3)	SMO (rad.)
99%	38/0	29/1	5/8	19/12	5/6	13/7	9/11	9/17	8/13	23/8
95%	38/0	30/1	5/9	19/12	7/6	14/7	9/15	9/18	9/14	24/9

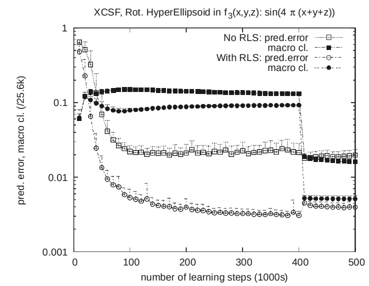
Piecewise Linear Function Approximation

- Conditions may be encoded as
  - Hyperrectangles
  - (Rotating) Hyperellipsoids
- Initialization, mutation, and crossover need to be adjusted
- Predictions as a linear function of the inputs
  - Gradient descent on weight vector ***or***
  - Recursive least squares approximation
- Evolves a partially overlapping piece-wise linear approximation

Performance in 3D Sinusoidal Function

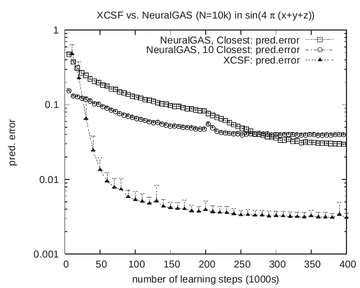
(Butz, Lanzi, & Wilson, in press)

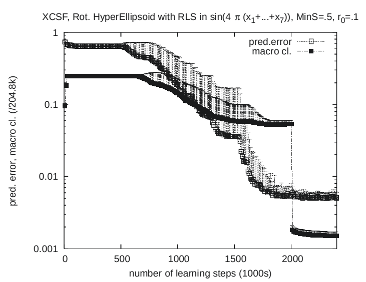




3D Function vs. Neural GAS, 7D Function with Compaction and CCM

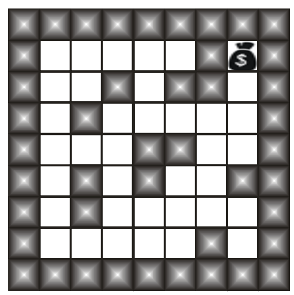
(Butz, Lanzi, & Wilson, in press)





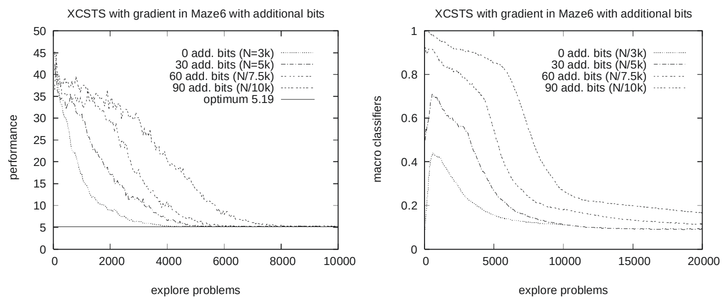
Performance in RL Problems Example: Maze6

- Reinforcement learning problems
  - Approximation of Q-value function
  - Reward propagation necessary
- MDP problems
- POMDP pose additional challenges (Lanzi, 2000; Lanzi, & Wilson, 2000)
- RL comparison in mountain-car problem (Lanzi, Loiacono, Wilson, & Goldberg, 2006)



Performance in Maze6 plus Irrelevant Bits

(Butz, Goldberg, & Lanzi, 2005; Butz, 2006)



Summary of XCS

- XCS is a highly flexible LCS.
- XCS can be applied to a variety of problem domains.
- XCS shows competitive or even superior performance.
- XCS generalizes well.
- XCS is noise robust.
- Further applications are imminent.

Anticipatory Learning Classifier Systems

1. Introduction
2. ACS2
3. XACS
4. Potentials

Anticipatory Learning Classifier Systems

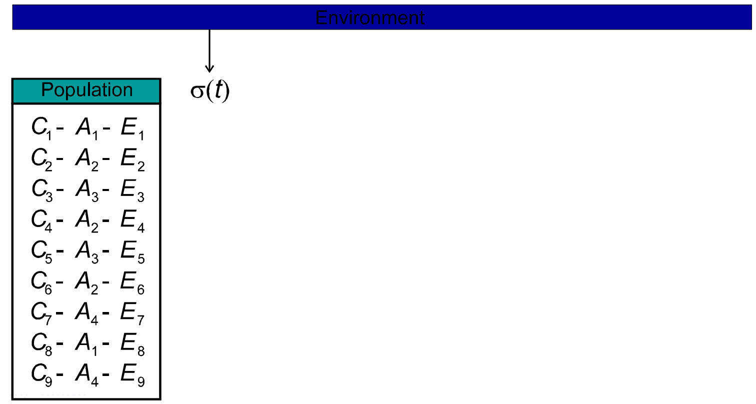
- Learning classifier systems (Michigan-style) that learn latently predictive world models (Riolo, 1991; Stolzmann, 1998).
- Each rule comprises a
  - Condition C,
  - Action A,
  - Effect part E,
  - Rule quality estimate F.
- Each rule explicitly predicts something like:
  Given condition C is satisfied and action A is executed, effect E is expected.
- Population represents a predictive environmental model.

ACS2: Rule Structure Learning

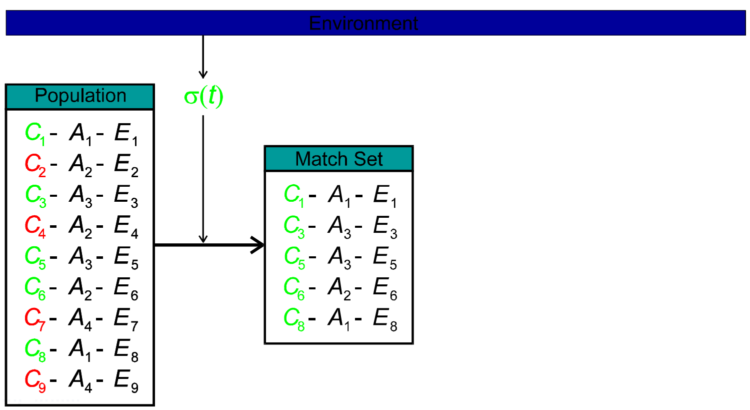
(Butz, 2002; Butz, Goldberg, & Stolzmann, 2002; Stolzmann, 1998)

- Anticipatory learning process
  - Primary learning of action-effect (R-E) relations
  - Secondary differentiation of conditions
  - *A directed, or informed, specialization mechanism*
- Genetic generalization mechanism
  - Fitness based on accuracy of effect-predictions
  - Selection of accurate classifiers
  - Deletion of inaccurate and/or highly specialized classifiers
  - *An undirected, genetic generalization mechanism*
- *ALP and GGM together evolve complete, accurate, and maximally general predictive models.*

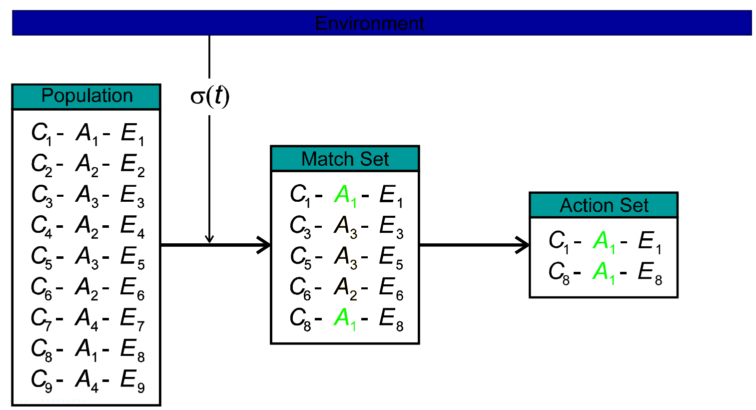
ACS2 – Problem Interaction



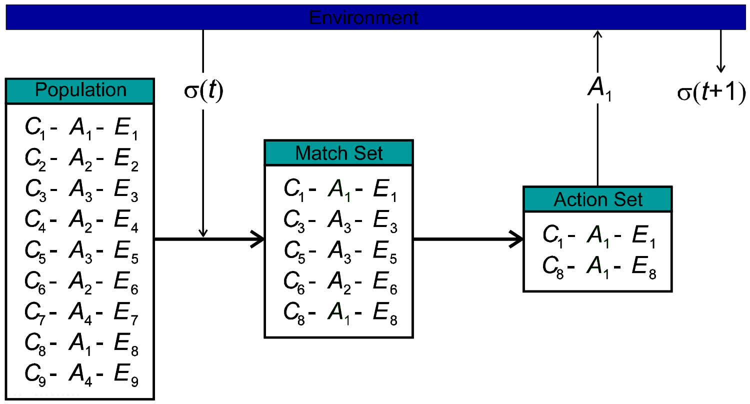
ACS2 – Match Set Formation



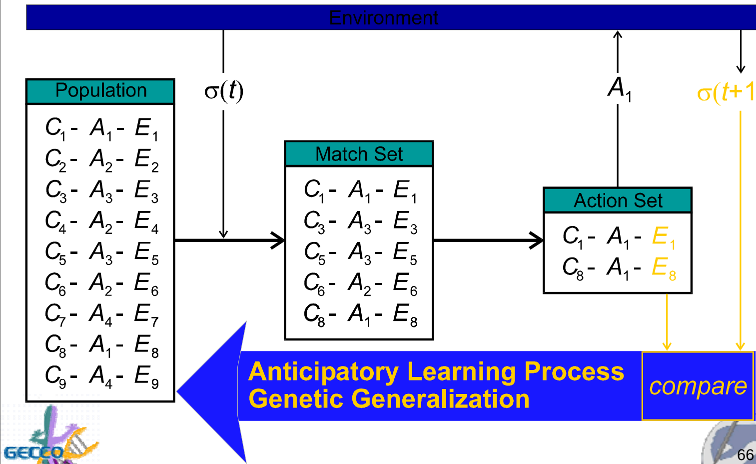
ACS2 – Action Set Formation



ACS2 – Action Execution



ACS2 – Learning

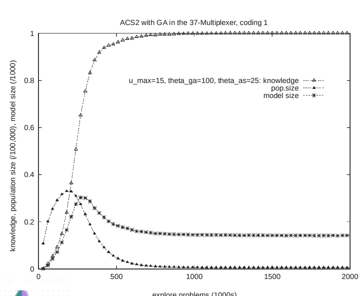


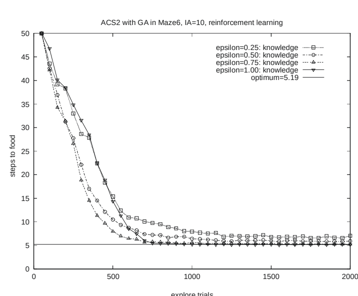
ACS2 – Performance Examples

(Butz, 2002)

Multiplexer performance: Class prediction

Maze 6 – Optimal behavior





Independent RL in ACS2 = XACS

- ACS2 represents reward prediction inside rules
  - RL directly in state-predictive rules.
  - But rule structure learning depends only on state prediction.
  - Can lead to model aliasing (model too general for accurate reward predictions)
- In XACS, behavior is realized in behavioral module
  - Learns generalized state values via XCS mechanism.
  - Model-based RL = online generalizing DYNA-PI mechanism (Sutton, 1990; Sutton, & Barto, 1998).

XCS as the State Value Learner

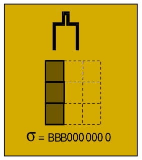
- XCS now approximates state values.
- Thus:
  - Population of classifiers with conditions only
  - Evaluation of classifiers by the means of ACS2
  - GA and fitness evaluation stay the same
- Updates of reward prediction in XCS via ACS2 predictions:

$$cl.V = (1-\beta)cl.V + \beta[P(\sigma)]$$

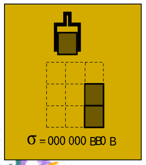
$$P(\sigma) = \rho(t) + \gamma \max_{\alpha} \frac{\sum_{cl\ \text{match}\ \text{Pred}(\sigma,\alpha,bg(\sigma,\alpha))} cl.V \cdot cl.\kappa \cdot cl.num}{\sum_{cl\ \text{match}\ \text{Pred}(\sigma,\alpha,bg(\sigma,\alpha))} cl.\kappa \cdot cl.num}$$

Example: Blocks World Problem

Goal



Current Situation



XACS predictive model population

Condition C	Action A	Effect E	q	Q-value (ACS only)
0## ### ### B	R1	B## ### ### 0	1.0	$.9^4$= .6561
### 0## ### B	R2	### B## ### 0	1.0	$.9^2 * .9^4 * .9^6$=.6658
### ### #B# B	R3	### ### ##B 0	1.0	$.9^6$= .5314
...				

State value XACS population

Condition C	V
##B ### ### #	1
#B# ### ### B	0.9
...	
B0# ### ### 0	$.9^4$= .6561
0## ### ### 0	$.9^6$= .5314
...	...

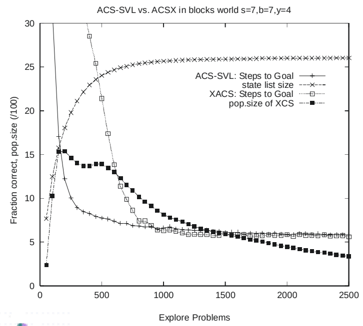
Current situation:
000 000 BB0 B

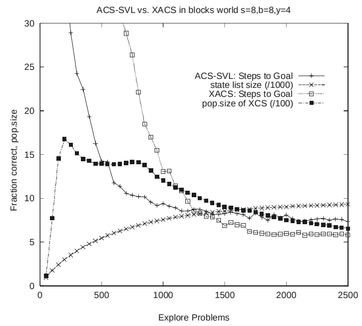
Predictions:
R1 → B00 000 BB0 0 → .656
R2 → 000 B00 BB0 0 → .531
R3 → 000 000 BBB 0 → .531

Resulting behavior:
Execute action R1

Blocks World Performance

(Butz, & Goldberg, 2003)





ACS2/XACS Potentials

- Learn generalized predictive model.
  - Fast and directed.
  - Currently restricted to mainly deterministic environments (irrelevant attributes may fluctuate).
  - May be enhanced with statistics-based specialization.
- Can be used to simulate cognitive phenomena
  - Anticipatory behavior in rats (Butz, & Hoffmann, 2002)
  - Motivational module available
  - Interactions of emerging motivations and emotions possible

ALCSs - Summary

- ALCSs are LCSs that learn generalized predictive world models online (latent learning).
- Behavioral policy is learned with state value learning mechanisms.
- Model-based reinforcement learning is possible.
- ACS2 – efficient predictive model learning
- XACS – online generalizing model and state value learning.
- Other ALCSs
  - YACS (Gérard, & Sigaud, 2001)
  - MACS (Gérard, Meyer, & Sigaud, 2005)

Other recent LCSs

1. ***Endogenous fitness approaches***
   – Economy- or energy-derived resource models for fitness estimations (Baum, 1999; Booker, 2000, 2001)
2. ***Genetic and artificial life environment*** (GALE)
   – A parallel, distributed Pitt-style GA (Llorà, Garrell, 2001; Bernadó, Llorà, Garrell, 2002).
3. ***Genetic classifier system (GAssist)***
   – Strongly generalizing Pitt-style datamining LCS (Bacardit, 2004).
4. ***Multiobjective LCS*** (MOLCS)
   – Multiobjective Pitt-style LCS (Llorà et al., 2003)
5. ***sUpervised Classifier System*** (UCS)
   – An XCS derivative for datamining problems (Bernadó-Mansilla, & Garrell-Guiu, 2003).

…and many others (see references).

Summary

- Learning Classifier Systems
  - Learn and generalize online (iteratively),
  - Extract useful problem sub-structures,
  - Combine gradient-based (rule evaluation) and evolutionary-based (rule structuring) learning techniques.
- LCSs represent their problem solutions by…
  … a set of (partially overlapping) sub-solutions (population of classifiers).
- LCSs can solve…
  - Classification problems (separation of problem classes)
  - Function approximation problems (piecewise approximation of function value)
  - Reinforcement learning problems (generalized Q-value function)
  - Other prediction problems (e.g. predictive environmental models)

Conclusions

- LCS is a very general and flexible learning paradigm.
  - Many condition and prediction representations are possible.
  - Many gradient-based learning mechanisms are possible.
  - Many rule discovery mechanisms are possible.
  - Other combinations and integrations of machine learning algorithms are possible.
- Thus:
  - Use the LCS most suitable for the problem at hand.
  - If necessary, optimize
    - Conditions (representation and evolution)
    - Predictions (representation and gradient-based approximation)

Further LCS Information

1. The LCS Web (Barry, 2007)
2. The LCS Bibliography (Kovacs, 2004)
3. Algorithmic descriptions of XCS and ACS2 (Butz, & Wilson, 2002; Butz, & Stolzmann, 2002).
4. LCS books and surveys: Butz (2002), Butz (2006), Bull (2004), Bull, & Kovacs (2005), Kovacs (2004), Sigaud, & Wilson (in press).
5. lcs-and-gbml Yahoo group (moderators: Xavier Llorà and John Holmes)
6. IWLCS proceedings (Lanzi, Stolzmann, & Wilson, 2000, 2001, 2002, 2003; Kovacs, Llorà, & Takadama, in press)
7. IWLCS 2007 workshop tomorrow (http://www.psychologie.uni-wuerzburg.de/i3pages/butz/IWLCS2007/)

References I

- Bacardit, J. (2004). Pittsburgh Genetic-Based Machine Learning in the Dame Mining Era: Representations, Generalization, and Run Time. *Computer Engineering Department, University of Ramon Llull,* Barcelona, Spain.
- Barry, A. (2007). The Learning Classifier Systems Web. University of Bath. http://lcsweb.cs.bath.ac.uk/.
- Baum, E. B. (1999). Toward a model of intelligence as an economy of agents. *Machine Learning*, 35, 155-185.
- Bernadó-Mansilla, E., & Garrell-Guiu, J.M. (2003). Accuracy-based learning classifier systems: Models, analysis, and applications to classification tasks. *Evolutionary Computation*,11, 209-238.
- Bernadó, E.; Llorà, X., & Garrell, J.M. (2002). XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks. In Lanzi, P.L.; Stolzmann, W. & Wilson, S.W. (Eds.), *Advances in Learning Classifier Systems (LNAI 2321)* (pp. 115-132). Berlin Heidelberg: Springer-Verlag.
- Bonarini, A. (2000). An introduction to learning fuzzy classifier systems. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Learning classifier systems: From foundations to applications (LNAI 1813)* (pp. 83-104). Berlin Heidelberg: Springer-Verlag.
- Booker, L. B. (1982). Intelligent Behavior as an Adaptation to the Task Environment. PhD thesis. *The University of Michigan*, Ann Arbor, MI.
- Booker, L. B. (1985). Improving the performance of genetic algorithms in classifier systems. In Grefenstette, J. (Ed.), *Proceedings of the 1st international conference on genetic algorithms and their application (ICGA-1985)* (pp. 80-92). Pittsburg, PA: Lawrence Erlbaum Associates.
- Booker, L. B. (2000). Do we really need to estimate rule utility in classifier systems? In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Learning classifier systems: From foundations to applications (LNAI 1813)* (pp. 125-142). Berlin Heidelberg: Springer-Verlag.

References II

- Booker, L. B. (2001). Classifier systems, endogenous fitness, and delayed rewards: A preliminary investigation. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, 921-926.
- Booker, L. B., Goldberg, D. E., & Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, 40, 235-282.
- Bull, L. (Ed.). Applications of learning classifier systems. Berlin Heidelberg: Springer-Verlag.
- Bull, L, & Hurst, J. (2002): ZCS redux. *Evolutionary Computation*, 10, 185-205.
- Bull, L. & Kovacs, T. (Eds.) (2005). *Foundations of learning classifier systems*. Berlin Heidelberg: Springer-Verlag.
- Bull, L., & O'Hara, T. (2002). Accuracy-based neuro and neuro-fuzzy classifier systems. *Proceedings of the Fourth Genetic and Evolutionary Computation Conference (GECCO-2002)*, 905-911.
- Butz, M. V. (2002). *Anticipatory learning classifier systems*. Kluwer Academic Publishers, Boston, MA.
- Butz, M. V. (2006). *Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design*. Studies in Fuzziness and Soft Computing Series, Springer Verlag, Berlin Heidelberg, Germany.
- Butz, M. V., Goldberg, D. E. & Lanzi, P. L. (2005). Computational complexity of the XCS classifier system. In Bull, L. & Kovacs, T. (Eds.), *Foundations of learning classifier systems* (pp. 91-126). Berlin Heidelberg: Springer-Verlag.
- Butz, M. V., Goldberg, D. E., & Lanzi, P. L. (2005). Gradient descent methods in learning classifier systems: Improving XCS performance in multistep problems. *IEEE Transactions on Evolutionary Computation,* 9, 452-473.
- Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2002). The anticipatory classifier system and genetic generalization. *Natural Computing*, 1, 427-467.

References III

- Butz, M. V., Goldberg, D. E., & Tharakunnel, K. (2003). Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11, 239-277.
- Butz, M. V., & Hoffmann, J. (2002). Anticipations control behavior: Animal behavior in an anticipatory learning classifier system. *Adaptive Behavior*, 10, 75-96.
- Butz, M. V., Kovacs, T., Lanzi, P. L., & Wilson, S. W. (2004). Toward a theory of generalization and learning in XCS. *IEEE Transactions on Evolutionary Computation*, 8, 28-46.
- Butz, M. V., Lanzi, P. L., & Wilson, S. W. (in press). Function Approximation with XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction. *IEEE Transactions on Evolutionary Computation.*
- Butz, M.V., & Stolzmann, W. (2002). An algorithmic description of ACS2. In Lanzi, P.L., Stolzmann, W., & Wilson, S.W. (Eds.). *Advances in learning classifier systems: Fourth international workshop, IWLCS 2001 (LNAI 2321)* (pp. 211-230). Berlin Heidelberg: Springer-Verlag.Butz, M. V., & Wilson, S. W. (2002). An Algorithmic Description of XCS. *Soft Computing*, 6, 144-153.
- Dixon, P. W., Corne, D. W., & Oates, M. J. (2003). A ruleset reduction algorithm for the XCS learning classifier system. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Learning classifier system: Fifth international workshop, IWLCS 2002 (LNAI 2661)* (pp. 20–29). Berlin Heidelberg: Springer-Verlag.
- Gérard, P., Meyer, J., & Sigaud, O. (2005). Combining latent learning and dynamic programming in MACS. *European Journal of Operational Research,* 160, 614-637.
- Gérard, P., & Sigaud, O. (2001). Adding a generalization mechanism to YACS. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001),* 951-957.
- Goldberg, D. E. (1983). Computer-Aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning. PhD thesis. *The University of Michigan*, Ann Arbor, MI.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization & machine learning*. Addison-Wesley.

References IV

- Holland, J.H. (1975). *Adaptation in natural and artificial systems.* University of Michigan Press.
- Holland, J.H. (1985). Properties of the bucket brigade algorithm. In Grefenstette, J. (Ed.), *Proceedings of the 1st international conference on genetic algorithms and their application (ICGA-1985)* (pp. 1-7). Pittsburg, PA: Lawrence Erlbaum Associates.
- Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction. Processes of inference, learning, and discovery*. MIT Press.
- Holland, J. H., & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D. A., & Hayes-Roth, F. (Eds.), *Pattern directed inference systems* (pp. 313-329). Academic Press.
- Kovacs, T. (1996). Evolving Optimal Populations with XCS Classifier Systems. Master thesis. *University of Birmingham*, Birmingham, UK.
- Kovacs, T. (2007). A Learning Classifier System Bibliography. Department of Computer Science, University of Bristol, UK. http://www.cs.bris.ac.uk/~kovacs/lcs/search.html.
- Kovacs, T. (2004). Strength of accuracy: Credit assignment in learning classifier systems. Berlin Heidelberg: Springer-Verlag.
- Kovacs, T., Llorà, X., & Takadama, K. (in press). Advances at the frontier of LCS (LNCS 4399). Berlin Heidelberg: Springer-Verlag.
- Lanzi, P. L. (1999). Extending the representation of classifier conditions. Part II: From messy coding to S-expressions. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, 345-352.
- Lanzi, P. L. (2000). Adaptive agents with reinforcement learning and internal memory. *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, 333-342.
- Lanzi, P. L., Loiacono, D., Wilson, S. W., & Goldberg, D.E. (2006). Classifier prediction based on tile coding. *GECCO 2006: Genetic and Evolutionary Computation Conference,* 1497-1504.

References V

- Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.) (2000). *Learning classifier systems: From foundations to applications (LNAI 1813)*. Berlin Heidelberg: Springer-Verlag.
- Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.) (2001). *Advances in learning classifier systems: Third international workshop, IWLCS 2000 (LNAI 1996)*. Berlin Heidelberg: Springer-Verlag.
- Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.) (2002). *Advances in learning classifier systems: 4th international workshop, IWLCS 2001 (LNAI 2321)*. Berlin Heidelberg: Springer-Verlag.
- Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.) (2003). *Learning classifier systems: 5th international workshop, IWLCS 2002 (LNAI 2661)*. Berlin Heidelberg: Springer-Verlag.
- Lanzi, P. L., & Wilson, S.W. (2000). Toward optimal classifier system performance in non-Markov environments. *Evolutionary Computation*, 8, 393-418
- Llorà, X., & Garrell, J. M. (2001). Knowledge independent data mining with fine-grained parallel evolutionary algorithms. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, 461-468.
- Llorà, X., Goldberg, D. E., Traus, I., & Bernadó, E. (2003). Accuracy, Parsimony, and Generality in Evolutionary Learning Systems via Multiobjective Selection. In Lanzi, P.L., Stolzmann, W. & Wilson, S.W. *(Eds.). Learning classifier systems: 5th international workshop, IWLCS 2002 (LNAI 2661)* (pp. 118-142). Berlin Heidelberg: Springer-Verlag.
- Riolo, R. L. (1991). Lookahead planning and latent learning in a classifier system. In Meyer, J. & Wilson, S.W. (Eds.). *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive* Behavior (pp. 316-326). *MIT Press.*
- Sigaud, O., & Wilson, S. W. (in press). Learning classifier systems: A survey. *Journal of Soft Computing.*
- Smith, S. F. (1980). A learning System Based on Genetic Adaptive Algorithms. PhD thesis, *University of Pittsburgh*, Pittsburgh, PA.

References VI

- Stolzmann, W. (1998). Anticipatory classifier systems, *Genetic Programming 1998: Proceedings of the Third Annual Conference,* 658-664.
- Sutton, R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning,* 216-224.
- Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. *MIT Press.*
- Valenzuela-Rendón, M. (1991). The fuzzy classifier system: A classifier system for continuously varying variables. Proceedings of the 4th International Conference on Genetic Algorithms (ICGA 1991), 346.353.
- Widrow, B., & Hoff, M. (1960). Adaptive switching circuits. *Western Electronic Show and Convention, part 4* (pp. 96-104). New York: Convention Record.
- Wilson, S. W. (1983). On the retino-cortical mapping. *International Journal of Man-Machine Studies*, 18, 361-389.
- Wilson, S. W. (1985). Knowledge growth in an artificial animal. In Grefenstette, J. (Ed.), *Proceedings of the 1st international conference on genetic algorithms and their application (ICGA-1985)* (pp. 16-23). Pittsburg, PA: Lawrence Erlbaum Associates.
- Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning*, 2, 199-228.
- Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2, 1-18.
- Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3, 149-175.
- Wilson, S. W. (2002). Compact rulesets from XCSI. In Lanzi, P.L., Stolzmann, W., & Wilson, S.W. (Eds.). *Advances in learning classifier systems: Fourth international workshop, IWLCS 2001 (LNAI 2321)* (pp. 196–208). Berlin Heidelberg: Springer-Verlag.