The Compact Classifier System: Scalability Analysis and First Results

Xavier Llorà, Kumara Sastry, & David E. Goldberg

Illinois Genetic Algorithms Lab
University of Illinois at Urbana-Champaign

{xllora,kumara,deg}@illigal.ge.uiuc.edu

Motivation

- Pittsburgh classifier systems
- Can we apply Wilson's ideas for evolving rule sets formed only by maximally accurate and general rules?
- Bottom up approach for evolving such rules
  - The compact classifier system
- Previous Multiobjective (Llorà, Goldberg, Traus, Bernadó, 2003) approaches were top down
  - Explicitly address accuracy and generality
  - Use it to push and product compact rule sets
- Side product:
  - Scalability challenge of De Jong & Spears (1991) representation

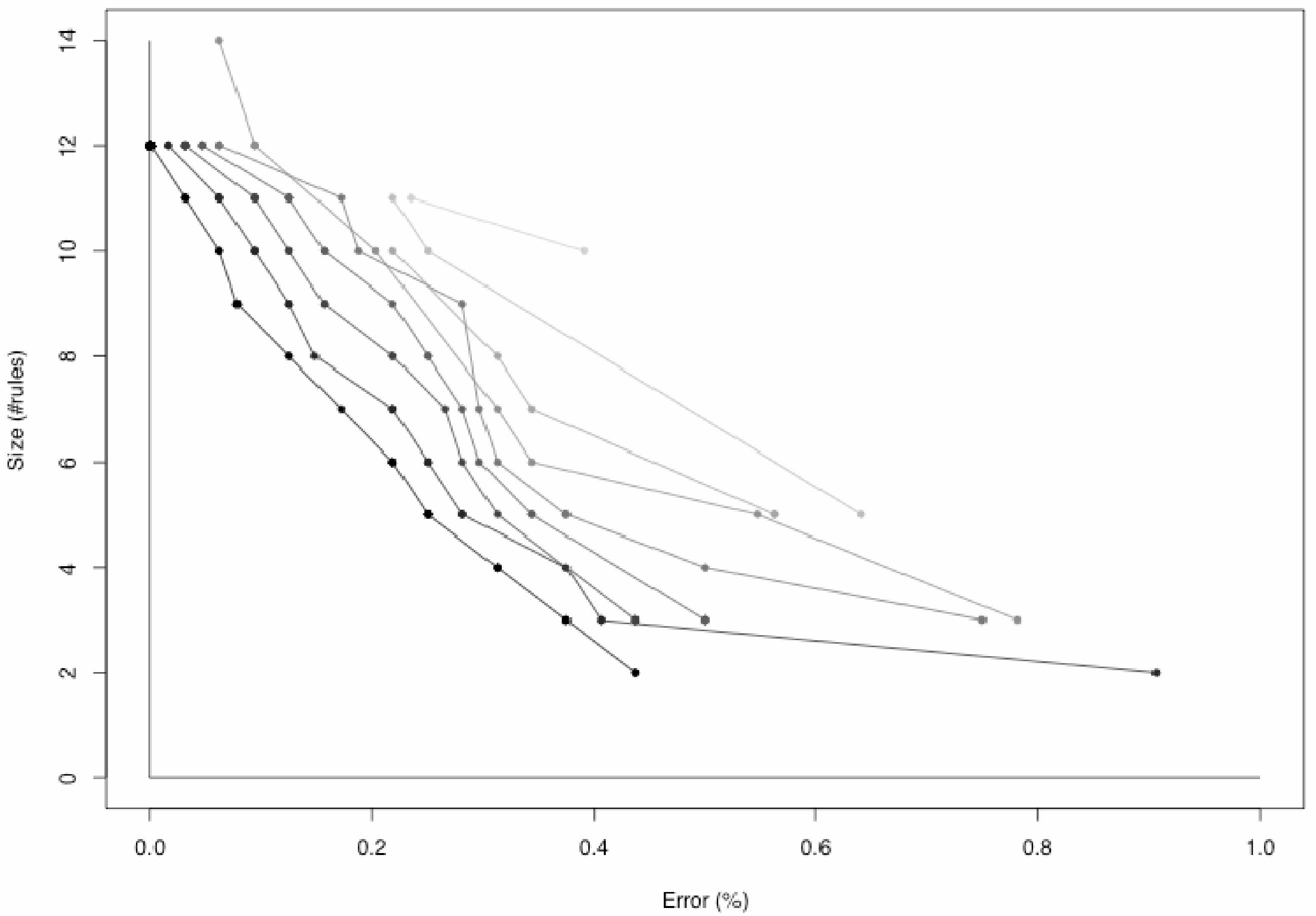
Binary Rule Encoding

- De Jong & Spears (1991)
- Widely used in Pittsburgh classifiers
- GALE, MOLS, GAssist have used it

color				*shape*		*size*			
red	**green**	**blue**	**white**	**round**	**square**	**huge**	**large**	**medium**	**small**
1	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**

- A rule is expressed as (**1111|01|0110**)
- Equivalent to Holland's (1975) representation (**#11,#12**)
- A rule set is a disjunction of such rules

Previous Efforts based using Multiobjective Optimization



(Llorà, Goldberg, Traus, Bernadó, 2003)

Maximally Accurate and General Rules

- Accuracy and generality can be computed using data set

$$\alpha(r) = \frac{n_{t+}(r) + n_{t-}(r)}{n_t} \qquad \varepsilon(r) = \frac{n_{t+}(r)}{n_m}$$

- Fitness should combine accuracy and generality

$$f(r) = \alpha(r) \cdot \varepsilon(r)^{\gamma}$$

- Such measure can be either applied to rules or a rule sets
- The *compact classifier systems* uses this fitness and a *compact genetic algorithm* (cGA) to evolve such rules
- Each cGA run use a different initial perturbed probability vector

The Compact Genetic Algorithm Can Make It

- Rules may be obtained optimizing

$$f(r) = \alpha(r) \cdot \varepsilon(r)^{\gamma}$$

- The basic cGA scheme
  1. Initialization $p_{x_i}^{0} = 0.5$
  2. Model sampling (two individuals are generated)
  3. Evaluation (*f(r)*)
  4. Selection (tournament selection)
  5. Probabilistic model updation
  6. Repeat steps 2-5 until termination criteria are met

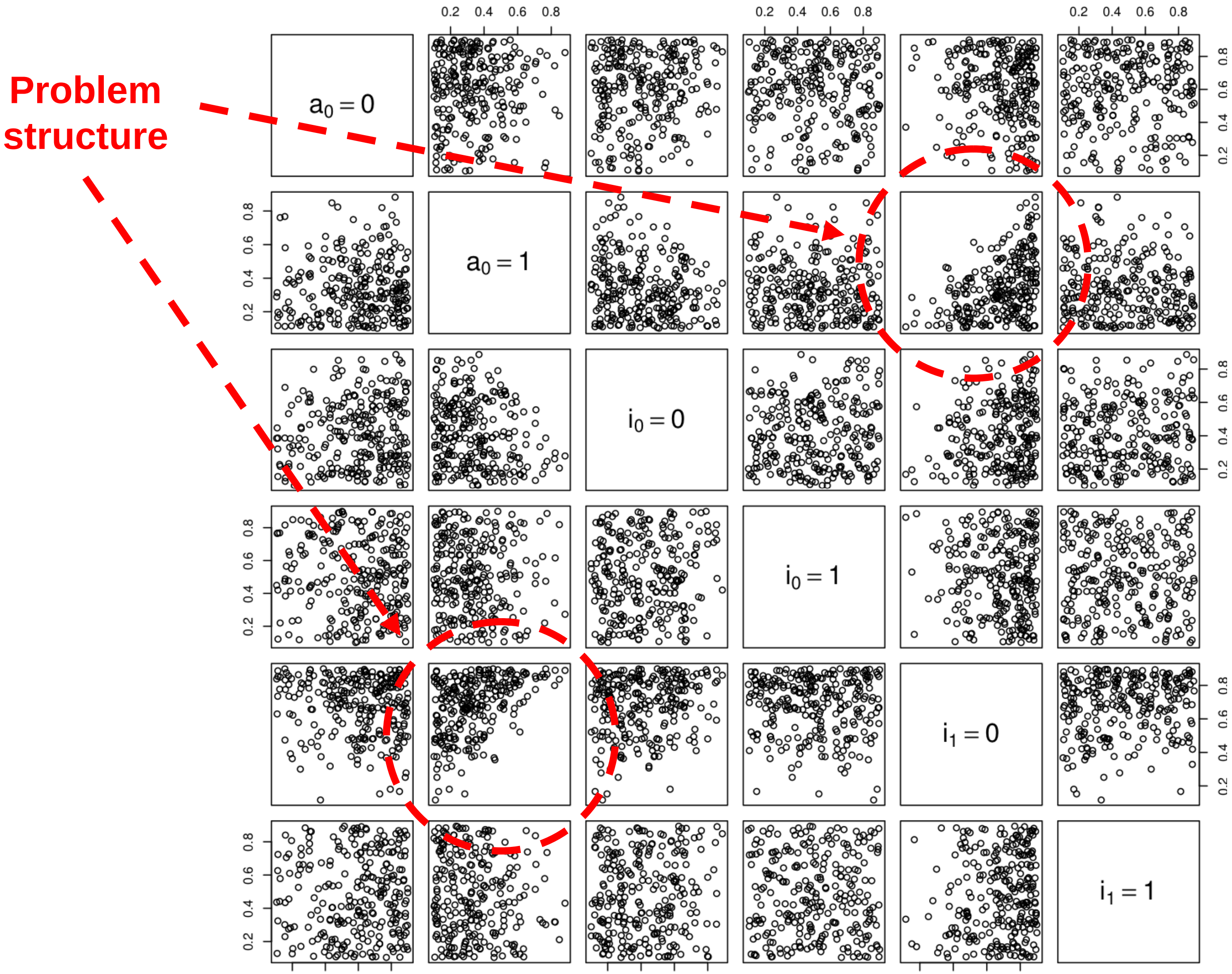
cGAModel Perturbation

- Facilitate the evolution of different rules
- Explore the frequency of appearance of each optimal rule
- Initial model perturbation

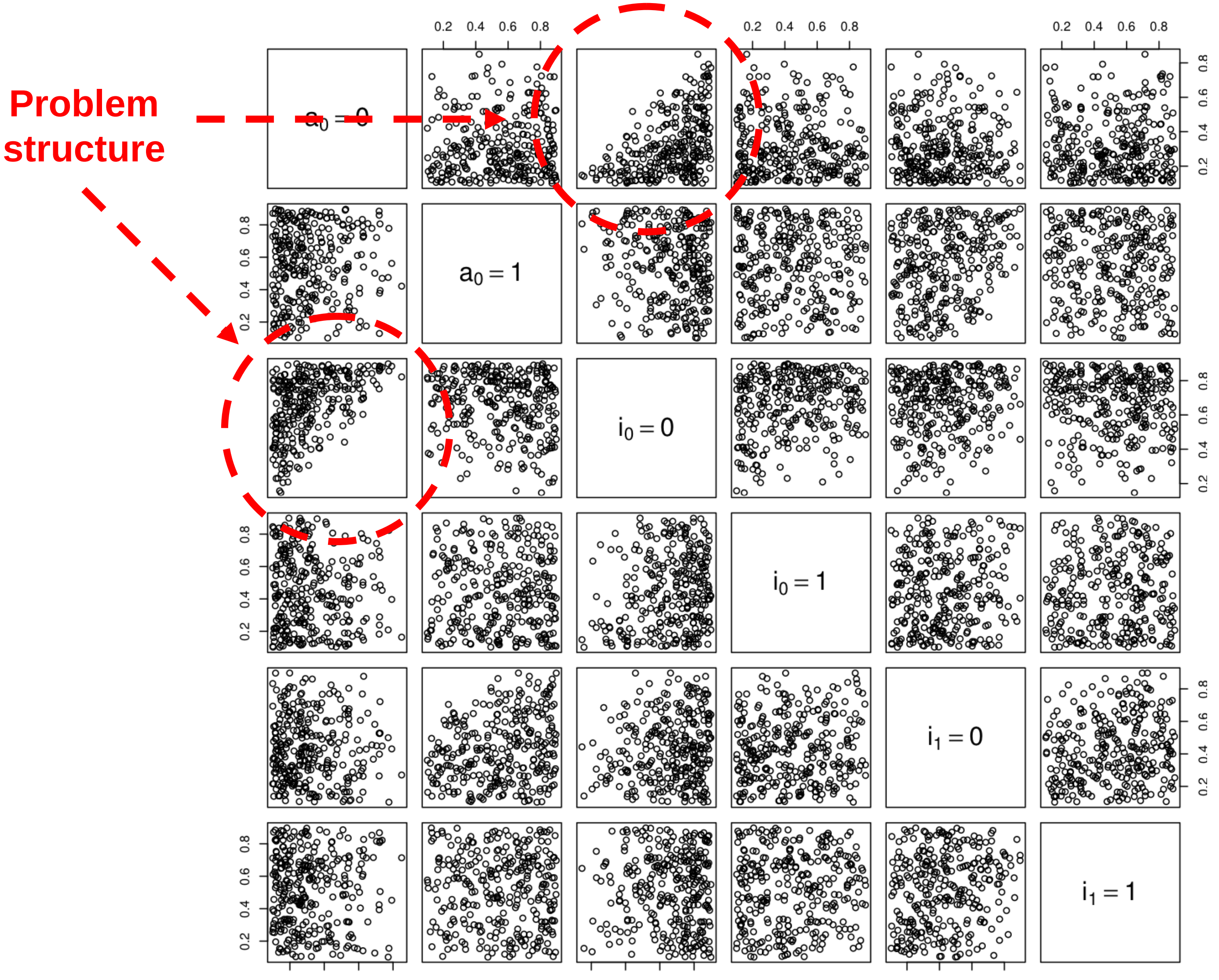
$$p_{x_i}^{0} = 0.5 + U(-0.4, 0.4)$$

- Experiments using the 3-input multiplexer
- 1,000 independent runs
- Visualize the pair-wise relations of the genes

Initial Perturbed Vectors Leading to rule 100111(01#)



Initial Perturbed Vectors Leading to rule 011101(1#1)



Perturbation Summary

- 97% of the runs lead to a maximally general and accurate rule
- The provability of evolving each of the optimal rules was roughly 1/3
- The initial perturbed probability vectors that lead to an optimal rule show pair-wise relations among genes
- The pair-wise relations reflect the problem structure

But One Rule Is Not Enough

- Model perturbation in cGA evolve different rules
- The goal: *evolve population of rules that solve the problem together*
- The fitness measure ($f(r)$) can be also be applied to rule sets
- Two mechanisms:
  - Spawn a population until the solution is meet
  - Fusing populations when they represent the same rule

Spawning and Fusing Populations of Rules

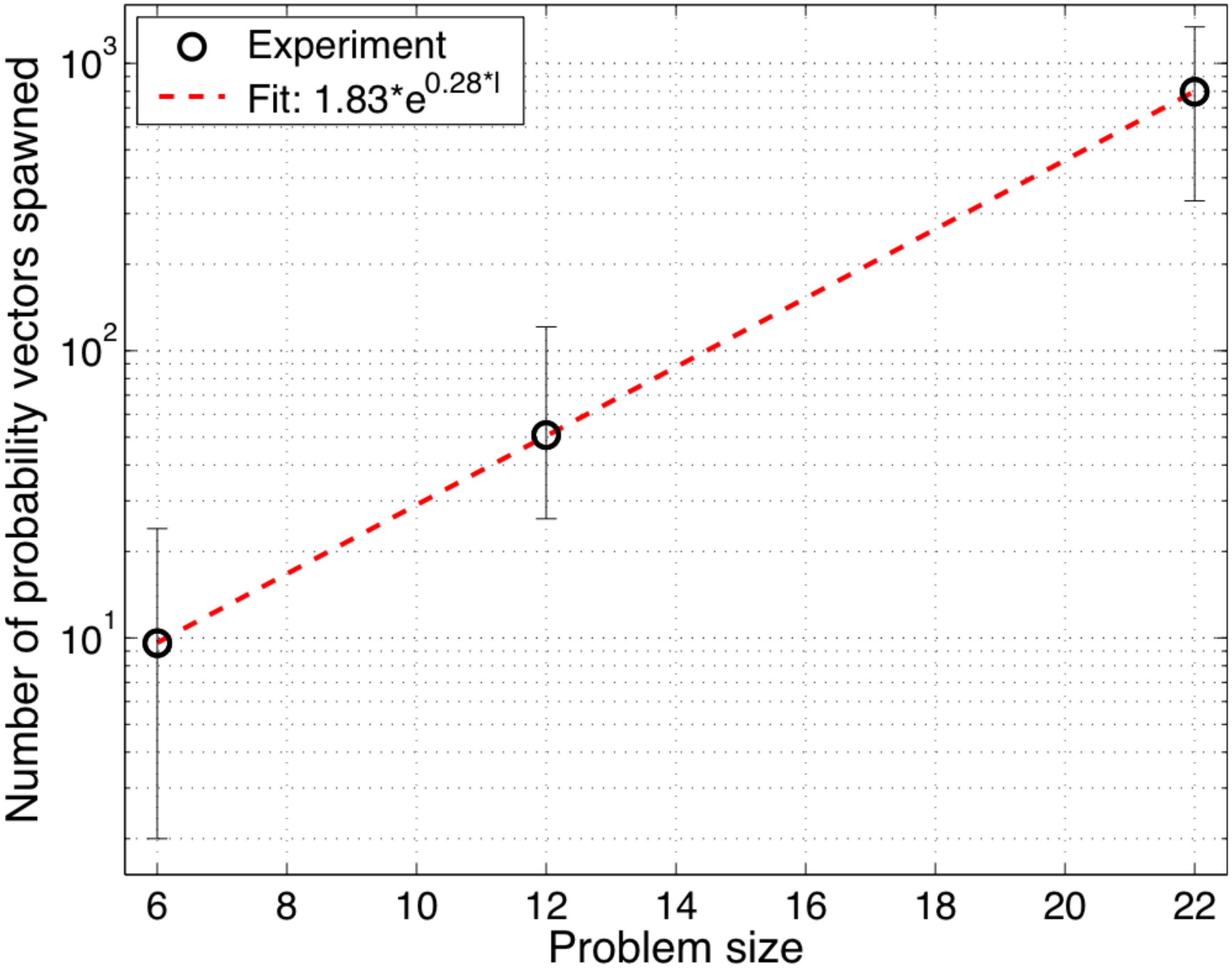
Table 1: Algorithmic description of the CCS.

1.	$\mathcal{D} \leftarrow \{pert(p_0), \ldots, pert(p_k)\}$.
2.	Foreach $p_i \in \mathcal{D}$ run cGA.
3.	$\mathcal{R} \leftarrow \{r_i$ sampled from $p_i\}$.
4.	Compute $f(\mathcal{R})$ using equation 3.
5.	If given $p_i, p_j \in \mathcal{D}$ and $d(p_i, p_j) < \theta$ then $\mathcal{D} \leftarrow \mathcal{D} \setminus \{p_i\}$.
6.	If $f(\mathcal{R}) = 1.0$ return $\mathcal{R}$ else $\mathcal{D} \leftarrow \mathcal{D} \cup \{pert(p)\}$ and goto 2.

Experiments & Scalability

- Analysis using multiplexer problems (3-, 6-, and 11-input)
- The number of rules in [O] grow exponentially
  - 2^i, where i is the number of inputs
- The CGA success as a function of the problem size
  - 3-input: 97%
  - 6-input: 73.93%
  - 11-input:43.03%
- Scalability over 10,000 independent runs

Scalability of CCS



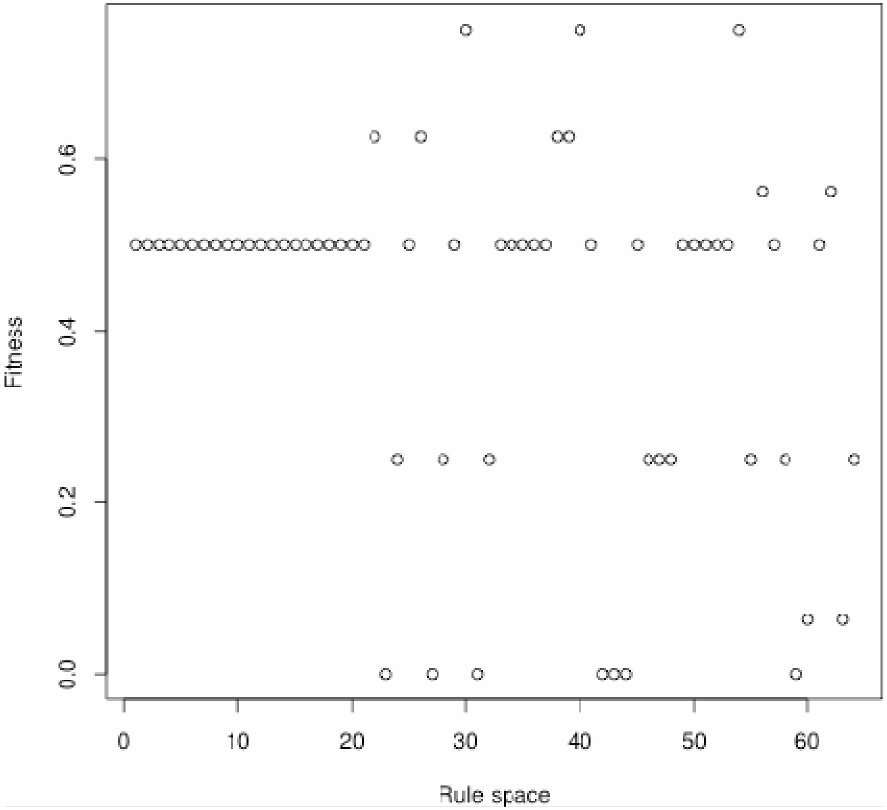
Unmatchable Rules: A Byproduct

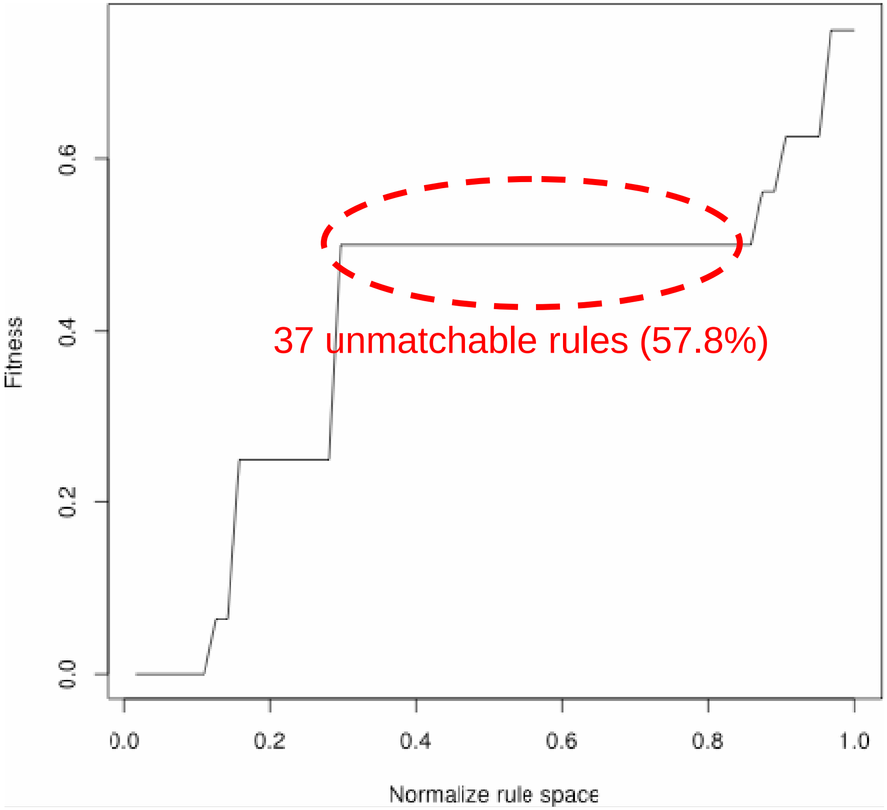
- A rule is unmatchable if:
  - At least one attribute in the contain have all its possible values set to 0

color				*shape*		*size*			
red	**green**	**blue**	**white**	**round**	**square**	**huge**	**large**	**medium**	**small**
1	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**

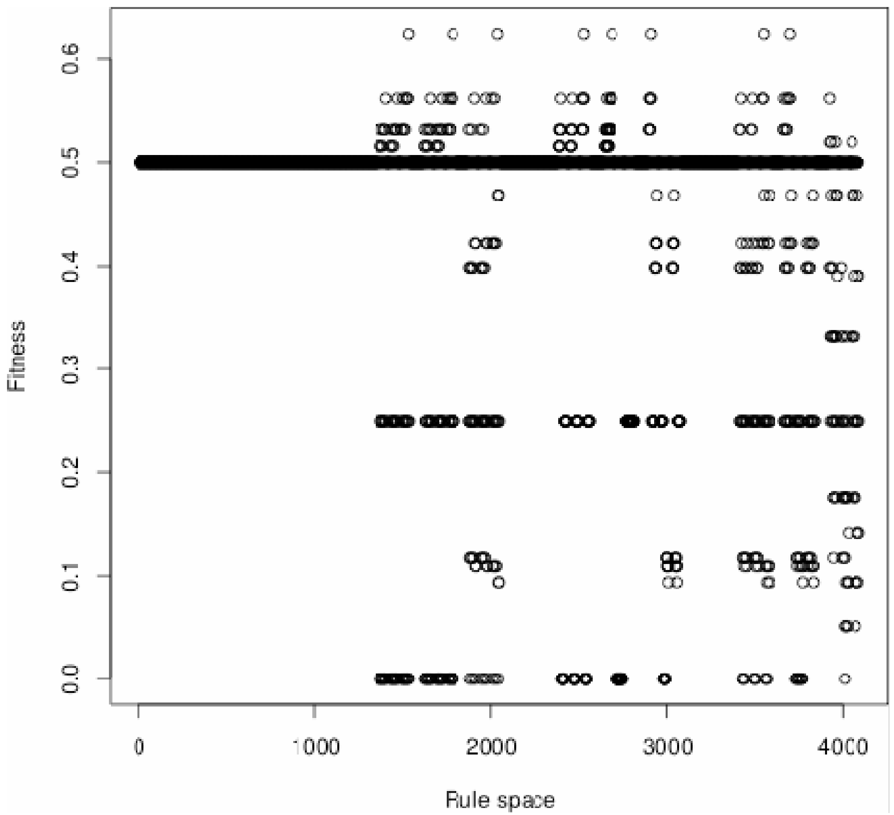
- The rule (**1111|00|0110**) force the shape to be neither round or square
- Hence no data instance will ever match it
- Direct impact on the scalability of LCS/GBML system using it (as simple experiments with the multiplexer show)

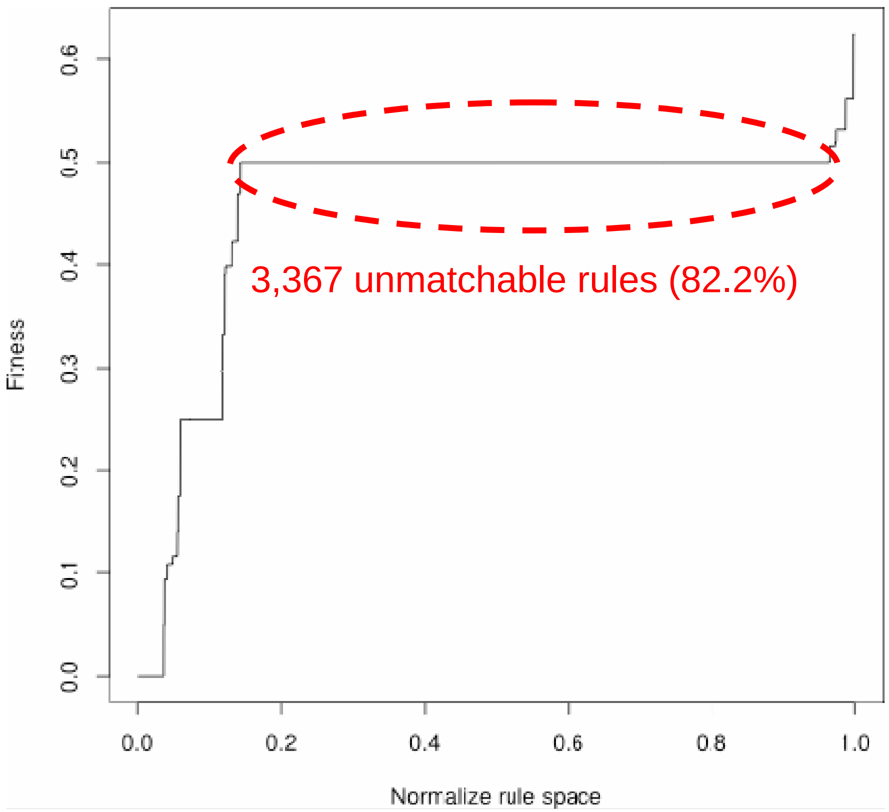
3-Input Multiplexer



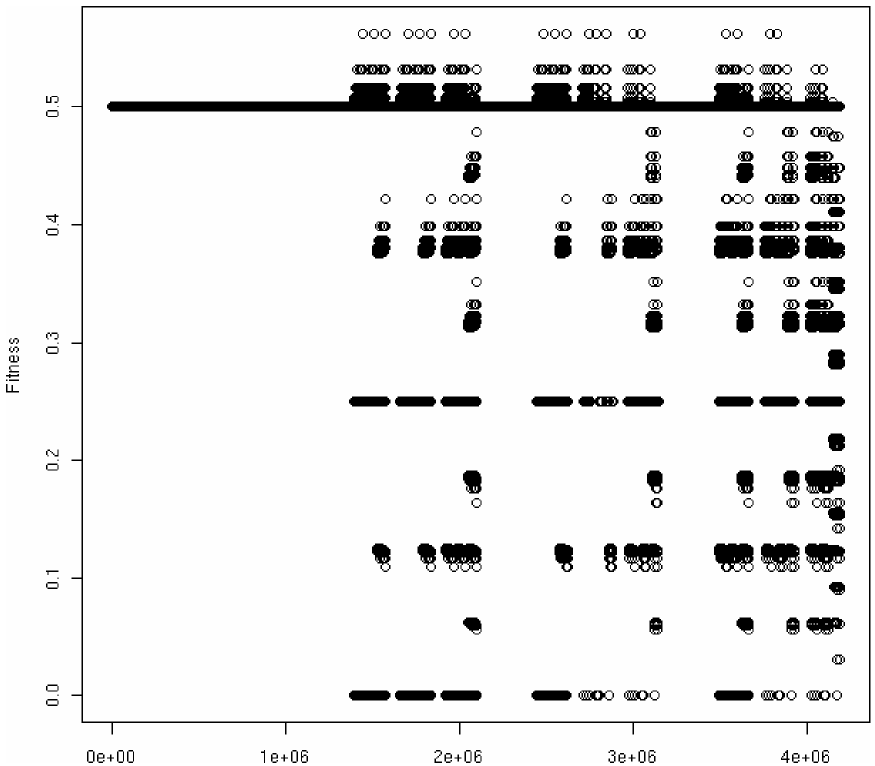


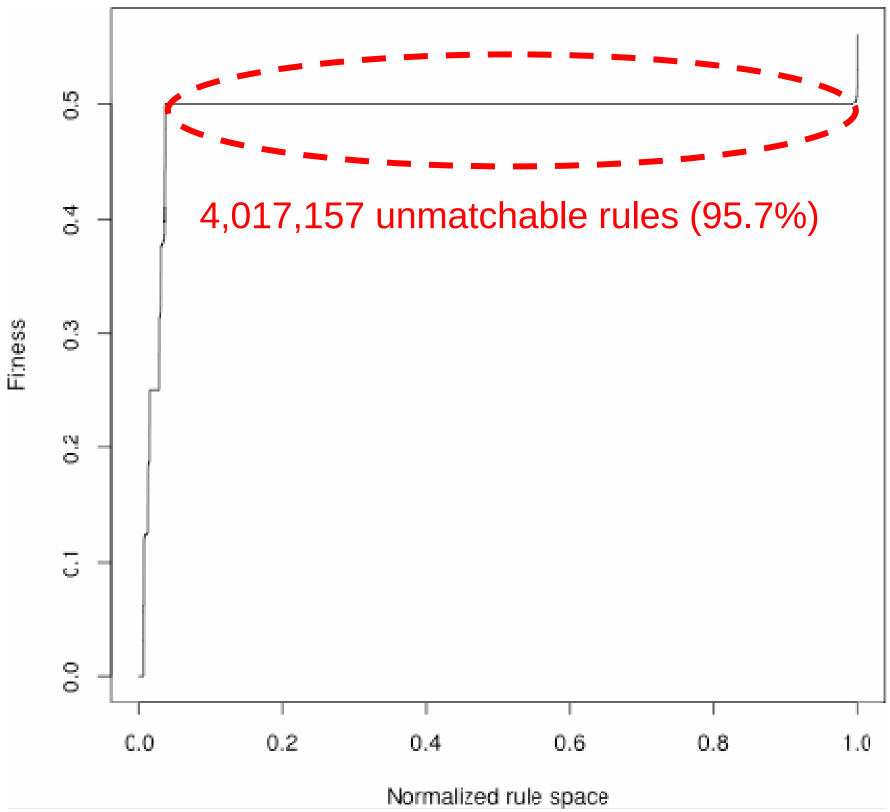
6-Input Multiplexer





11-Input Multiplexer





Growth Ratio of Unmatchable Rules (I/III)

- An unmatchable rule has of all attribute values set to 0
- Analysis for problems with binary attributes (worst case)
- The total number of rules

$$\Sigma(l) = 2^l$$

- Number of rules matchable rules (all attributes set to either 01, 11, & 11)

$$\Psi(l) = 3^{\frac{l}{2}}$$

- Size of the unmatchable rule set plateau

$$\Phi(l) = \Sigma(l) - \Psi(l) = 2^l - 3^{\frac{l}{2}}$$

Growth Ratio of Unmatchable Rules (II/III)

- Growth ratio of unmatchable rules

$$\rho(l) = \frac{\Phi(l)}{\Psi(l)} = \frac{2^l}{3^{\frac{l}{2}}} - 1$$

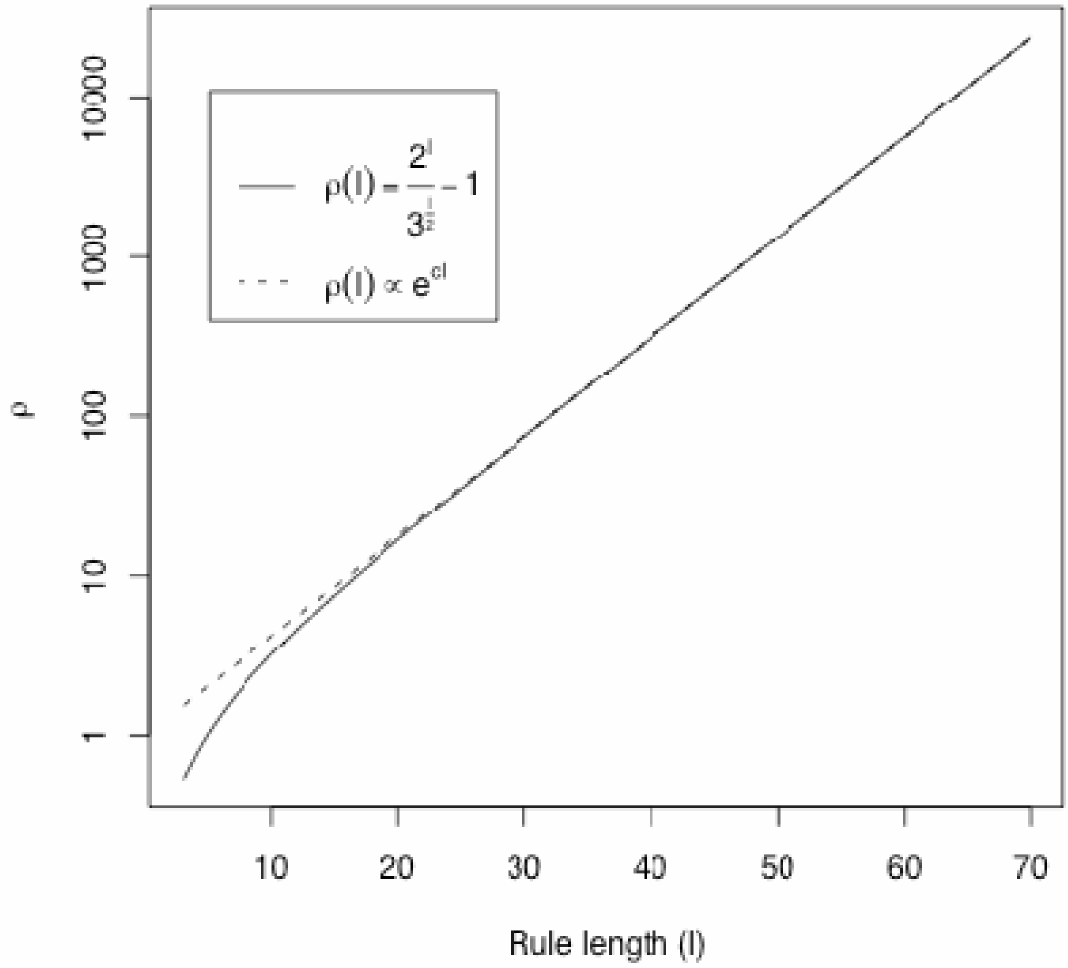
- It can be approximated by

$$\rho(l) \approx e^{cl}$$

$$c = \ln\left(\frac{2}{\sqrt{3}}\right) = 0.143$$

- The growth ratio (ρ) for this representation grows exponentially

Growth Ratio of Unmatchable Rules (III/III)



Conclusions

- Initial steps to evolve rule sets formed formed only by maximally accurate and general rules using Pittsburgh systems
- Using a cGA and the appropriate fitness function (CCS) we can evolve such rules
- Rule representation has a direct connection to the scalability of any GBML system
  - A wrong choice makes the problem extremely hard
- Further analysis for different representations is needed (Stone, 2004)