Surrogate Fitness via Factorization of Interaction Matrix

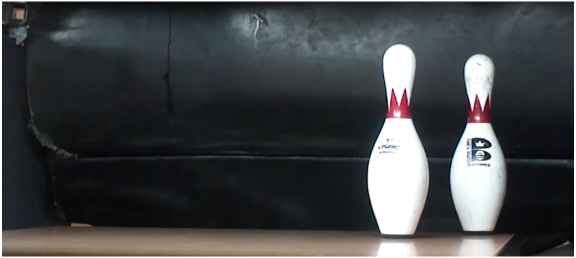
Paweł Liskowski   Krzysztof Krawiec

Computational Intelligence Group
Institute of Computing Science
Poznan University of Technology, Poland

March 30, 2016

Thought experiment

Thought experiment



Fitness Evaluation

Observation 1

GP algorithms do not let programs know **which** tests they solve.

Typical fitness function in GP *aggregates* program's behavior on *tests* by

- counting the number of passed tests (discrete domains).

$$f(p) = |\{y_i \neq \hat{y}_i(p)\}|_i \tag{1}$$

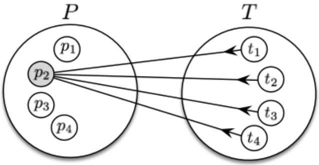
- summing the errors on individual tests (continuous domains).

$$f(p) = \sum_i (y_i - \hat{y}_i(p))^2 \tag{2}$$

Test-based Genetic Programming

Observation 2

Detailed information on particular interactions with **is available** in GP.



$g : P \times T \to [0,1]$

G	t_1	t_2	t_3	t_4
p_1	1	1	0	0
p_2	0	1	0	1
p_3	1	1	1	0
p_4	0	1	1	0

- P: set of m programs,
- T: set of n tests (fitness cases)
- $g(p,t)$: interaction function between $p \in P$ and $t \in T$
- G: $m \times n$ matrix of interaction outcomes between P and T
- Test-based problems (Pollack, Bucci, de Jong, Popovici)

Fitness Evaluation

Observation 3

Interaction outcomes for particular tests are *partially* dependent.

Can be used to derive **alternative search objectives** (*search drivers*):

- *Derivation of Search Objectives*
  (Krawiec & Liskowski, EuroGP 2015, ECJ 2016)

Fitness Evaluation

Observation 3

Interaction outcomes for particular tests are *partially* dependent.

Can be used to derive **alternative search objectives** (*search drivers*):

- *Derivation of Search Objectives*
  (Krawiec & Liskowski, EuroGP 2015, ECJ 2016)

Hypothesis

- An interaction outcome $g(p, t)$ can be reconstructed from other elements of G
- We may **reduce** so the number of program-test interactions.

The idea

Matrix factorization to estimate some program-test interactions in G.

SFIMX: Surrogate Fitness via Factorization of Interaction Matrix

Algorithm

1. Calculate **sparse** interaction matrix G between P and T
   - For each $p \in P$ draw a random subset of $\alpha|T|$ tests $T' \subset T$
   - Apply p to tests in T'
2. **Factorize** G into non-negative components W and H (rank $\leq k$)
3. **Reconstruct** the interaction outcomes by calculating $\hat{G} = WH$

- $\alpha \in (0, 1]$ - desired density of partial interaction matrix

Example: $P = \{p_1, \ldots, p_4\}$, $T = \{t_1, \ldots, t_5\}$, $\alpha = \frac{3}{5} = 0.6$

$$G = \begin{array}{c c} & \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} & \begin{pmatrix} 2 & & 1 & 2 & \\ & 2 & 1 & & 1 \\ 1 & & & 2 & 2 \\ 2 & 1 & & & 1 \end{pmatrix} \end{array}$$

Missing outcomes due to $\alpha < 1$

Example

$$
G = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array}
\begin{array}{c}
\begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\
\begin{pmatrix} 2 & & 1 & 2 & \\ & 2 & 1 & & 1 \\ 1 & & & 2 & 2 \\ 2 & 1 & & & 1 \end{pmatrix}
\end{array}
$$

Missing outcomes due to $\alpha < 1$

$$
W = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array}
\begin{array}{c}
\begin{array}{ccc} f_1 & f_2 & f_3 \end{array} \\
\begin{pmatrix} 0.46 & 1.96 & 0.6 \\ 1.27 & 0.1 & 0.95 \\ 1.37 & 0.02 & 2.83 \\ 0.4 & 1.86 & 1.60 \end{pmatrix}
\end{array},
\quad
H = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array}
\begin{array}{c}
\begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\
\begin{pmatrix} 0.48 & 1.50 & 0.01 & 0.41 & 0.41 \\ 0.87 & 0.14 & 0.19 & 0.77 & 0.01 \\ 0.11 & 0.09 & 1.02 & 0.50 & 0.51 \end{pmatrix}
\end{array}
$$

$$
\hat{G} = WH = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array}
\begin{array}{c}
\begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\
\begin{pmatrix} 2 & 1.02 & 1 & 2 & 0.52 \\ 0.8 & 2 & 1 & 1.07 & 1 \\ 1 & 2.31 & 2.1 & 2 & 2 \\ 2 & 1 & 2.01 & 2.4 & 1 \end{pmatrix}
\end{array}
\xrightarrow{f(p_i) = \sum_{j=1}^{n} g_{ij}}
\begin{array}{l} f(p_1) = 6.54 \\ f(p_2) = 5.87 \\ f(p_3) = 9.41 \\ f(p_4) = 8.41 \end{array}
$$

- Sizes of W and H controlled by parameter $k \geq 1$ (here: $k = 3$)
- Technical realization: multiplicative update rule.
- Cost of evaluation **reduced** $\frac{1}{\alpha}$ **times.**

Experiment

- GP: 'vanila' GP, $|P| = 1000$
- SFIMX: $|P|$ increased $(1 - \alpha)$ times $\implies$ same budget
  - FULL: $k = |T|$,
  - HALF: $k = \frac{|T|}{2}$,
  - LOG: $k = log_2|T|$
- RSS: Calculates fitness using $\alpha|T|$ random tests (same budget)
- $\alpha \in 0.1, 0.2, \ldots, 1.0$

Domain	Instruction set	Problem	Variables	#tests
Boolean	and, nand or, nor	Cmp6	6	64
		Cmp8	8	256
		Par5	5	32
		Mux6	6	64
		Maj6	6	64
Algebras	$a_i(x, y)$	Disc-a1...a5	3	27
	$a_i(x, y)$	Malcev-a1...a5	3	15

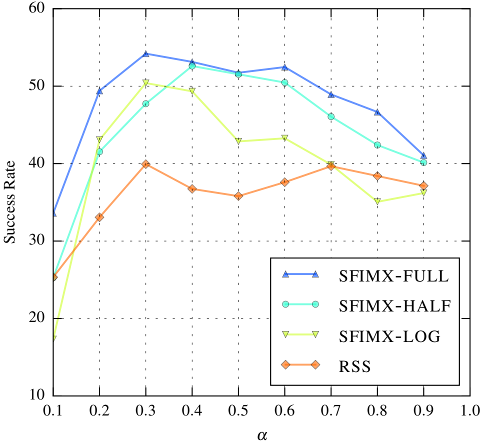
Experiment

Average ranks on success rate (Friedman's $p \ll 0.001$)

	SFIMX			GP	RSS
	HALF	FULL	LOG		
All problems	2.07	2.13	2.67	3.90	4.23
Boolean	2.4	1.7	3.3	3.2	4.4
Categorical	1.90	2.35	2.35	4.25	4.15

- Best results for $\alpha = 0.3$ and $\alpha = 0.4$
- Roughly the same performance as GP using only 10% of interactions
- LOG variant $\rightarrow$ high compression without affecting the performance
  - $\Longrightarrow$ Interaction outcomes are indeed corrrelated.
- Low k $\rightarrow$ low computational overhead of factorization
  - For SFIMX-LOG: only 6% of the total cost of $1,000|T|$ interactions

Impact of α on success rates



Success rates improve as sparsity in G increases up to $\alpha = 0.3$

Conclusions

- SFIMX = well-informed and scalable surrogate fitness.
- Target domains:
  - Problems with expensive interaction functions
  - Problems with large numbers of tests
  - Evolving controllers, two-player games, image analysis, ...
- Replaces a discrete fitness function with a continuous one.
- Ongoing work: continous domains

Conclusions

- SFIMX = well-informed and scalable surrogate fitness.
- Target domains:
  - Problems with expensive interaction functions
  - Problems with large numbers of tests
  - Evolving controllers, two-player games, image analysis, ...
- Replaces a discrete fitness function with a continuous one.
- Ongoing work: continous domains

Thank You