Behavioral Program Synthesis: Insights and Prospects

Krzysztof Krawiec[1], Jerry Swan[2], Una-May O'Reilly[3]

[1]Poznan University of Technology, Poland
[2]University of York, UK
[3]Massachusetts Institute of Technology, USA

Genetic Programming Theory and Practice (GPTP) 2015
University of Michigan, Ann Arbor

May 15, 2015

Agenda

- Bottlenecks
- Blackboxes
- Whiteboxes

Evaluation bottleneck

Limited informativeness of objective functions

Example: Synthesize 11-bit multiplexer

- Objective function $f : \mathbb{S} \to [0, 2048]$
- Minimal potential solution: a program tree with 11 leaves
- $C_{10}4^{10}11! \approx 7 \times 10^{17}$ potential solutions for instruction set {AND, NAND, OR, NOR} (C_n - Catalan number)

Search process navigates in a space of 10^{17} candidate solutions, using 11 bits of information per candidate solution.

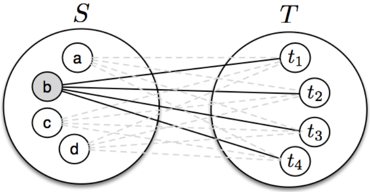
Consequence:
Poorly informed search algorithm.

Evaluation bottleneck

- More detailed information on solution's 'behavior' is often available.
- What is behavior?
- Behavior = the outcome of solution's interaction with multiple:
  - tests (GP)
  - initial conditions (control problems)
  - environments (behavioral/evolutionary robotics)
  - opponents (games)
  - problem instance (hyperheuristics)
- Formally: interactive domains, test-based problems.

Test-based problems

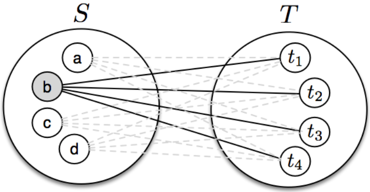
- Candidate solutions $\mathbb{S}$
- Tests $\mathbb{T}$
- Interaction function $g : \mathbb{S} \times \mathbb{T} \to \mathbb{R}$ (payoff function, loss function)
- Interaction matrix G - $m \times n$ between $S \subset \mathbb{S}$ S and $T \subset \mathbb{T}$



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

G	t_1	t_2	t_3	t_4
a	1	1	0	0
b	0	1	0	1
c	1	1	1	0
d	0	1	1	0

- Candidate solutions $\mathbb{S}$
- Tests $\mathbb{T}$
- Interaction function $g : \mathbb{S} \times \mathbb{T} \rightarrow \mathbb{R}$ (payoff function, loss function)
- Interaction matrix G - $m \times n$ between $S \subset \mathbb{S}$ S and $T \subset \mathbb{T}$



$g : S \times T \rightarrow [0, 1]$

G	t_1	t_2	t_3	t_4
a	1	1	0	0
b	0	1	0	1
c	1	1	1	0
d	0	1	1	0

Why aggregate?

How to widen the bottleneck?

Theory:

- Fitness-distance correlation
- Elementary fitness landscapes

'Unstructured' approaches: focus on diversity and hardness of tests)

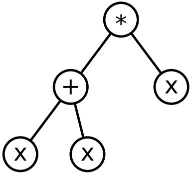
- Implicit fitness sharing (Smith *et al.* 1993; McKay 2000)
- Co-solvability (Krawiec & Liskowski 2010)
- Lexicase selection (Helmuth & Spector 2014)

'Structured' approaches: focus on (presumed) problem structure

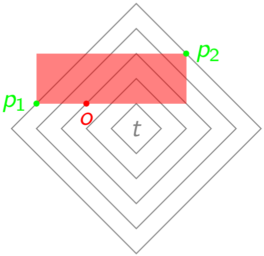
- Semantic GP
- Discovery of underlying objectives
- Behavioral GP

Avenue 1: Semantic-aware search operators

Program p:



x_i	$p(x_i)$
-0.5	**0.5**
1.0	**2.0**
1.5	**4.5**
2.0	**8.0**




```
                               AND
                      T1  =   /   \
                            X1     X2
        OR                                                   OR
       /    \                                               /   \
   AND        AND              OR                        AND      X3
  /   \      /   \     T2  =  /  \          T3  =       /   \
T1     TR  NOT    T2        X2    X3                 AND     NOT
            |                                       /   \     |
            TR                NOT                 X1     X2   X3
                       TR  =   |
                               X3
```

(McPhee et al. 2007; Krawiec & Lichocki 2009; Moraglio, Krawiec, Johnson 2012)

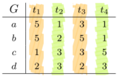
Avenue 2: Heuristic discovery of underlying objectives

$S = \{a, b, c, d\}$ $\quad T = \{t_1, t_2, t_3, t_4\}$

G	t_1	t_2	t_3	t_4
a	5	1	3	1
b	5	2	5	1
c	1	3	3	5
d	2	3	2	3

derived objectives t_{1+3} and t_{2+4}

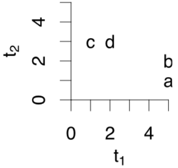
G'	t_{1+3}	t_{2+4}
a	4	1
b	5	1.5
c	2	4
d	2	3

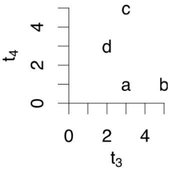


G	t_1	t_2	t_3	t_4
a	5	1	3	1
b	5	2	5	1
c	1	3	3	5
d	2	3	2	3

objective compression

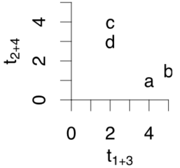
$b \succ a$





performance on tests t_1 and t_3 is correlated

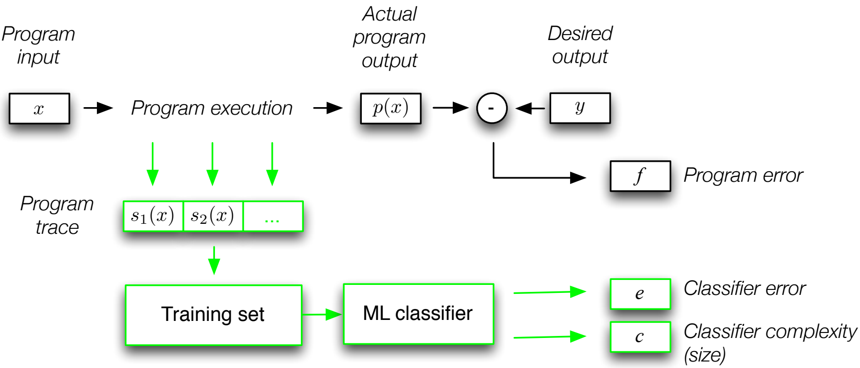
$b \succ a$ and $c \succ d$



some information about dominance is lost

(Krawiec and Liskowski 2014, 2015)

Avenue 3: Behavioral Evaluation



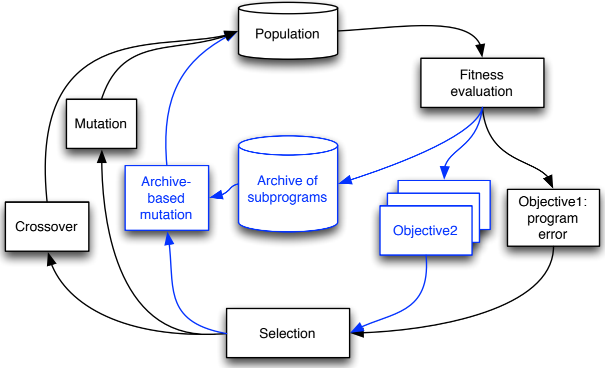
- **Black:** Conventional GP
- **Green:** Pattern-guided EA (PANGEA)

(Krawiec and Swan 2013, Krawiec & O'Reilly 2014, Krawiec & Solar-Lezama 2014)

Avenue 3: Behavioral Evaluation

Behavioral programming = PANGEA extended with:

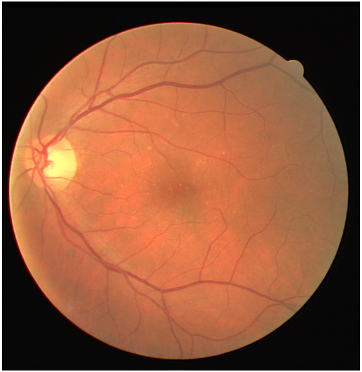
- Multiobjective evaluation and selection (NSGA-II, Deb *et al.* 2002),
- Subprograms indicated by classifier archived and reused in mutation.

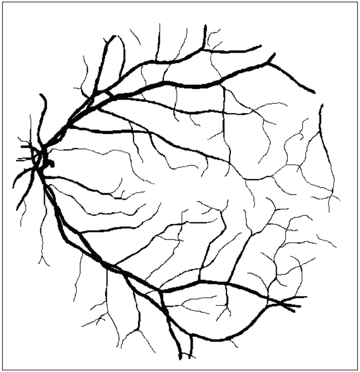


(Krawiec and Swan 2013, Krawiec & O'Reilly 2014)

Application: Detection of blood vessels in fundus images

Ongoing research on Optical Coherence Tomography (OCT) imaging
GP evolves classifiers (feature detectors) that work with BRIEF-like features.





Training image (left) and the corresponding manual segmentation (right).

Beyond evaluation bottleneck

Evaluation bottleneck is only one of manifestations of 'domain barrier' dogma.

- Particularly in GP, problem formulation is rich in domain-specific knowledge about formal properties of a problem.

Beyond evaluation bottleneck

Evaluation bottleneck is only one of manifestations of 'domain barrier' dogma.

- Particularly in GP, problem formulation is rich in domain-specific knowledge about formal properties of a problem.

Example: The power of types.
If the signature of the function to be synthesized is

$$f : \mathrm{List}[T] \rightarrow \mathbb{N},$$

then $f(x)$ has to be a function of length of x (*Theorems for free* (Wadler 1989))

This type of knowledge can be exploited: *Gen-O-Fix, Polyfunic, Hylas* (Swan et al. 2013-2015)

Open questions

- How much structure is in there?
    - Is discovering that structure worth the effort?
- Claim: There is a lot of structure to be discovered.
    - Real-world problems are structured by the math and physics of our Universe.
    - Even more in GP: The structure partially stems from the programming language used for synthesis.
- Real-world problems are more structured than we think.
    - Maths is structuring evaluation, dependencies between variables, etc.

Conclusions

Take-home messages:

- Objective functions = provide unbiased performance measure, not to **drive** search process.
- Open the blackboxes where possible
- Abandon scalar evaluation

Consequences:

- Better performance.
- Insight into problems.
- Richer design space for other components of metaheuristics

Questions?

ScEVO & ScaPS (Scala for Automated Program Synthesis)

Generic iterative metaheuristic:

```
def apply[S <: State](step: S => S)(stop: Seq[S => Boolean]): S => S = {
  @tailrec def iterate(s: S): S = stop.forall((sc: S => Boolean) => !sc(s)) match {
    case false => s
    case true  => iterate(step(s))
  }
  iterate
}
```

Semantic geometric crossover:

```
trait GeometricCrossoverL2 {
  this : Randomness =>
  def apply(tree1: Op, tree2: Op): List[Op] = {
    val a = rng.nextDouble
    List(Op("+",
      Op("*", Op(a), tree1),
      Op("*", Op(1 - a), tree2)))
  }
}
```

Generalized Evaluation and Search Drivers

Objective functions = designed to provide unbiased performance measure, not to **drive** search process.

Generalized Evaluation:

1. Evaluation function: $eval : \mathbb{S} \to \mathbb{E}$
   - Evaluation = any formal object that may help driving search
   - E.g., entire interaction matrix, set of program traces,
2. Search driver $f : \mathbb{E} \to \mathbb{O}$, where $\mathbb{O}$ is a partially ordered set.

Properties of search drivers:

- Contextual, qualitative, non-stationary, not extremalized at optima, weak, ...
- To be used along with other search drivers.
- Not the same as surrogate fitness!

Conventional objective function = special case of search driver