Behavioral Program Synthesis for the Automated Design of Algorithms

6th Workshop on Evolutionary Computation for the Automated Design of Algorithms

Krzysztof Krawiec

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

July 21, 2016

Outline

1. What is the problem and how did we get here?
2. A few directions to solve the problem.
3. A broader perspective - **search drivers** and **behavioral program synthesis**.

Program/algorithm synthesis

- The goal: Efficient synthesis of programs/algorithms
  - expression trees,
  - fully-fledged programs,
  - hyperheuristics, etc.
- Program = an executable structure that can interact with data
- Problem specification = set of examples
  - tests in GP
  - problem instances in ADA

An *iterative* search problem:

- Needs ways of prioritizing search
- The common means: (scalar) objective function
  - E.g., the number of passed tests/solved instances

Downsides of conventional objective functions

- The right way to assess the objective quality of solutions,
- ... but not designed to *drive* the search.
- Predicated on the "big valley" assumption: search moves tend to lead to similarly-valued solutions
- Very minimalist.

Downsides of conventional objective functions

- The right way to assess the objective quality of solutions,
- ... but not designed to *drive* the search.
- Predicated on the "big valley" assumption: search moves tend to lead to similarly-valued solutions
- Very minimalist.

Example: 6-bit multiplexer, $2^6 = 64$ tests:

- Number of possible fitness values: $2^6 + 1 = 65$ (≈ 6 bits)
- Number of possible 'output behaviors': $2^{64} = 1.84 \times 10^{19}$ (64 bits)
- Number of possible programs: far greater.

Downsides of conventional objective functions

- The right way to assess the objective quality of solutions,
- ... but not designed to *drive* the search.
- Predicated on the "big valley" assumption: search moves tend to lead to similarly-valued solutions
- Very minimalist.

Example: 6-bit multiplexer, $2^6 = 64$ tests:

- Number of possible fitness values: $2^6 + 1 = 65$ (≈ 6 bits)
- Number of possible 'output behaviors': $2^{64} = 1.84 \times 10^{19}$ (64 bits)
- Number of possible programs: far greater.

Evaluation bottleneck

Consequences

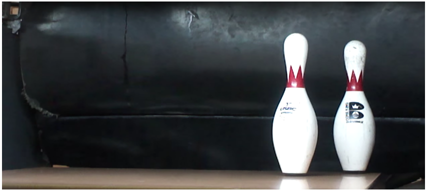
Consequences:

- Compensation: programs that pass different tests obtain same fitness.
- Passing all tests rewarded equally.

Consequences

Consequences:

- Compensation: programs that pass different tests obtain same fitness.
- Passing all tests rewarded equally.



Why stick to objective functions?

Objective reasons:

- Elegant and convenient
- Universal, 'plug&play' interface to many search/optimization methods
- Sometimes the only source of information on the problem available
    - Black-box optimization, IP restrictions, ...
    - However, not in GP and ADA.

Why stick to objective functions?

Objective reasons:

- Elegant and convenient
- Universal, 'plug&play' interface to many search/optimization methods
- Sometimes the only source of information on the problem available
  - Black-box optimization, IP restrictions, ...
  - However, not in GP and ADA.

Subjective reasons:

- Routine and legacy
- Human urge to linearly order/rank solutions

Why stick to objective functions?

Objective reasons:

- Elegant and convenient
- Universal, 'plug&play' interface to many search/optimization methods
- Sometimes the only source of information on the problem available
  - Black-box optimization, IP restrictions, ...
  - However, not in GP and ADA.

Subjective reasons:

- Routine and legacy
- Human urge to linearly order/rank solutions

Doing away with the bottleneck

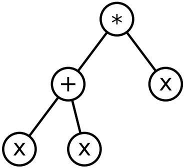
Behavioral Program Synthesis:

1. Obtain **more information** on solution's characteristics.
2. Elicit **alternative information** on solution's characteristics.
3. Design **search operators** capable of exploiting that information

Some 'avenues':

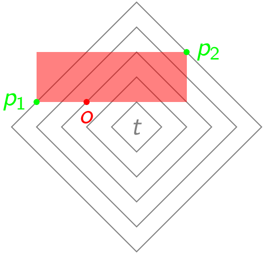
1. Semantic GP
2. Exploitation of interaction matrices
3. Behavior-based characterization

Program



p:

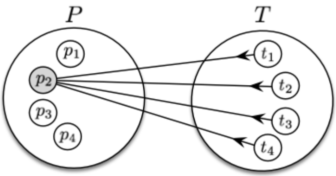
x_i	$p(x_i)$
-0.5	**0.5**
1.0	**2.0**
1.5	**4.5**
2.0	**8.0**




```
                                        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: GP as a test-based problem



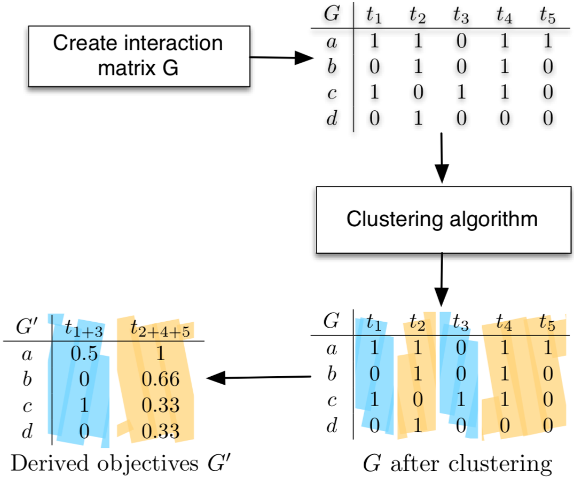
$$g : P \times T \rightarrow [0, 1]$$

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)

The idea: extract some alternative/additional information from G

2.1: DOC: **D**iscovery of Search **O**bjectives by **C**lustering[1]



G	t_1	t_2	t_3	t_4	t_5
a	1	1	0	1	1
b	0	1	0	1	0
c	1	0	1	1	0
d	0	1	0	0	0

G	t_1	t_2	t_3	t_4	t_5
a	1	1	0	1	1
b	0	1	0	1	0
c	1	0	1	1	0
d	0	1	0	0	0

G after clustering

G'	t_{1+3}	t_{2+4+5}
a	0.5	1
b	0	0.66
c	1	0.33
d	0	0.33

Derived objectives G'

[1] Paweł Liskowski and Krzysztof Krawiec. "Discovery of Implicit Objectives by Compression of Interaction Matrix in Test-Based Problems". In: *Parallel Problem Solving from Nature – PPSN XIII*. ed. by Thomas Bartz-Beielstein et al. Vol. 8672. Lecture Notes in Computer Science. Heidelberg: Springer, 2014, pp. 611–620. ISBN: 9783319107615

2.2: Non-negative matrix factorization (NMF)

Given G, find W and H such that

$$G \approx WH \;\; s.t. \;\; W, H \geq 0,$$

or more precisely:

$$\min_{W,H} f(W,H) \equiv \frac{1}{2}||G - WH||_F^2 \;\; s.t. \;\; W, H \geq 0,$$

- Effective, gradient-based algorithms exist
- Widely used in machine learning (recommender systems)

NMF: Example 1

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

$$W \times H = \begin{array}{c c} & \begin{array}{cc} f_1 & f_2 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} & \begin{pmatrix} 0.70 & 2.05 \\ 0.73 & 0.66 \\ 0.35 & 1.02 \end{pmatrix} \end{array} \times \begin{array}{c c} & \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{array}{c} f_1 \\ f_2 \end{array} & \begin{pmatrix} 0.70 & 0.70 & 2.70 & 2.70 \\ 0.74 & 0.74 & 0.06 & 0.06 \end{pmatrix} \end{array}$$

NMF: Example 2

$$G' = \begin{array}{c c} & \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} & \begin{pmatrix} 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ {\color{blue}2} & 1 & 1 & 1 \end{pmatrix} \end{array}$$

$$W' = \begin{array}{c c} & \begin{array}{cc} f_1 & f_2 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} & \begin{pmatrix} 0.96 & 1.51 \\ 0.39 & 1.84 \\ 0.86 & 0.38 \end{pmatrix} \end{array}, \quad H' = \begin{array}{c c} & \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{array}{c} f_1 \\ f_2 \end{array} & \begin{pmatrix} 2.16 & 1.20 & 0.72 & 0.72 \\ 0.05 & 0.35 & 0.90 & 0.90 \end{pmatrix} \end{array}$$

$$W' \times H' = \begin{array}{c c} & \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} & \begin{pmatrix} 2.17 & 1.70 & 2.07 & 2.07 \\ 0.95 & 1.13 & 1.96 & 1.96 \\ 1.88 & 1.17 & 0.97 & 0.97 \end{pmatrix} \end{array}$$

DOF: **D**iscovery of Search **O**bjectives by **F**actorization

The algorithm:

1. Calculate the interaction matrix G between S and T.
2. Factorize G into W and H
3. Define the derived objectives g'_j based on W and H, e.g.,

$$f_j(p) = w_{pj}$$

4. Use g'_js for **multiobjective evaluation/selection**.

SFIMX: Surrogate Fitness via Factorization of Interaction Matrix[2]

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

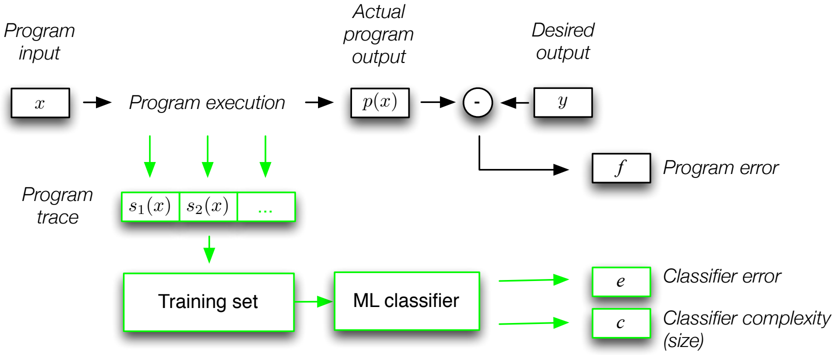
Missing outcomes due to $\alpha < 1$

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

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

[2] Pawel Liskowski and Krzysztof Krawiec. "Surrogate Fitness via Factorization of Interaction Matrix". In: *EuroGP 2016: Proceedings of the 19th European Conference on Genetic Programming*. Ed. by Malcolm I. Heywood et al. Vol. 9594. LNCS. Porto, Portugal: Springer Verlag, 30.03–1.04.2016, pp. 65–79.

Avenue 3: Behavioral Evaluation[3]



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

[3] Krzysztof Krawiec, Jerry Swan, and Una-May O'Reilly. "Behavioral Program Synthesis: Insights and Prospects". In: *Genetic Programming Theory and Practice XIII*. ed. by Rick Riolo, Jason H. Moore, and Mark Kotanchek. Genetic and Evolutionary Computation. Ann Arbor, USA: Springer, 14-16 05 2016. DOI: 10.1007/978-3-319-34223-8_10.

Programs are behaviorally rich

and so do **search and optimization algorithms**.

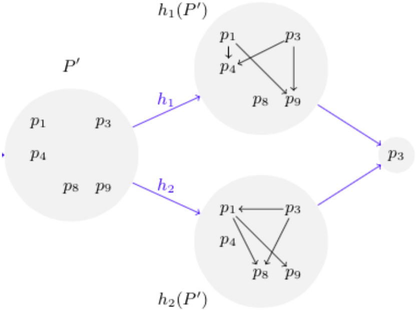
- Interaction outcome = algorithm's performance on a problem instance
- Execution trace = search trajectory
- ...

See: The Metaheuristics in the Large (MitL) initiative (Swan et al. 2014)

Unified conceptual framework?[4]

Search driver = a measure *designed to guide* the search process.

Objective function	Search driver
global	local
complete	partial
absolute	relative
context-free	contextual
stationaly	non-stationary



Multiple 'weak' search drivers rather than **one 'strong'** objective

[4] Krzysztof Krawiec. *Behavioral Program Synthesis with Genetic Programming*. Vol. 618. Studies in Computational Intelligence. Springer, 2016.

Open questions

- Are my search drivers consistent with the objective function?
- 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.
- Real-world problems are more structured than we think.
  - Maths is structuring evaluation, dependencies between variables, etc.

Conclusions

Take-home messages:

- Objective functions = not necessarily designed to **drive** search process.
- **Open** the bottlenecks and blackboxes!
- Consider abandoning objective functions in favor **search drivers**.

Potential gains:

- Better performance
- Additional insight into problems
- Still quite universal
- Richer design space for other components of metaheuristics

Thank You