Open Issues in Genetic Programming

Michael O'Neill, Leonardo Vanneschi,
Steven Gustafson & Wolfgang Banzhaf

UCD, Universidade Nova de Lisboa,
GE Global Research, Memorial University of Newfoundland
m.oneill@ucd.ie, lvanneschi@isegi.unl.pt,
steven.gustafson@research.ge.com, banzhaf@cs.mun.ca

http://www.sigevo.org/gecco-2013/

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

Instructors/Presenters

- **Michael O'Neill**

- **Leonardo Vanneschi**

- **Steven Gustafson**

- **Wolfgang Banzhaf**

Tutorial Outline

- Introduction & Motivation
- Open Issues in GP
- Questions & Discussion

Introduction & Motivation

O'Neill M., Vanneschi L., Gustafon S., Banzhaf W. (2010).
Open Issues in Genetic Programming.
Genetic Programming and Evolvable Machines 11(3-4):339-363
http://www.springerlink.com/content/a058142636361453/fulltext.pdf

++

Introduction

- GP in 30 seconds
- Why isn't GP more popular?
- Some stats & interesting results
- Historic time for GP with rise of Data Science
- Objective of tutorial
    - Identify roadblocks
    - Suggest future areas of research

GP in 30 Seconds

- Method for modeling data or learning behavior - regression models or robot control

Generate New Solutions w/ bias*

Data & Params

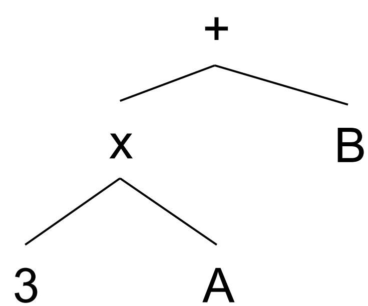
Output best “solution”

Evaluate and Rank Solutions

* First set of solutions usually “randomly” created

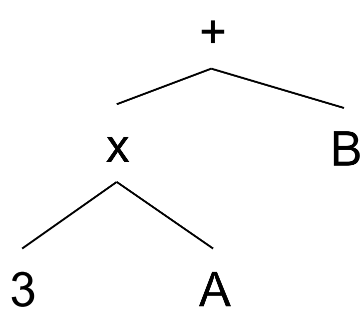
GP in 30 Seconds

- Search over solution structure & parameters



GP in 30 Seconds

- Input way of measuring success

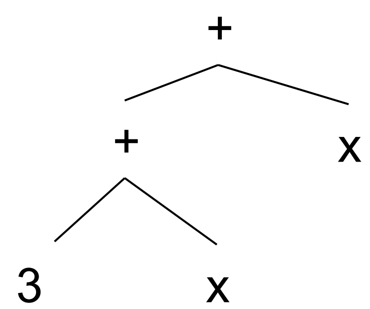


Solution A

Y	x	Solution A result
7	2	7
1	3	9
0	1	5
4	6	15

GP in 30 Seconds

- Input things a solution are built out of



% * sin() 3.14 7 9 x^(t-1)

GP in 30 Seconds

- Decide when to stop

After so many solutions sample

After no new good solutions found

After a solution of a given quality is found

Why Open Issues? Who cares?

- It is pretty popular!

- Publications, Conferences, Results in broad fields
  - Bioinformatics
  - Regression modeling
  - Robotics, and many others! (we've been to space!) ?

- How to raise the bar?
- Stochastic methods are often second choice due to lack of theory and predictable space, time complexity

Why Open Issues? Who cares?

- And… good for the community to step back and think where we need help

- And… we are at a historic time…

Historic time with Data Science

- Software, data and analytics driving new revolution in Industry, Gov't and Society!
- Data science postings and tools exploding
- Lipson named an Influential data science by a Forbes study due to GP work (Science paper)
- Data science right in GP best (SG's opinion) application: symbolic regression!

Historic time with Robots

- DARPA Humanoid challenge
- Human assisted bots (please clean my house)
- Advanced manufacturing
- More controls and cybernetics than we “probably” can scale in current way

Objective of Tutorial

- Identify and discuss issues
- Suggest possible new research areas
- Interaction and discussion

- As long members of community, we feel a bit like GP-startup-employees, we want it to succeed! Help us make it succeed!

Open Issues

1. Generalization STEVE
2. Complexity STEVE
3. Representations MIKE
4. Modularity   MIKE
5. Dynamic problems MIKE
6. Open-ended evolution MIKE
7. Fitness landscapes and problem difficulty LEO
8. Semantics LEO
9. Influence of Biology WOLFGANG
10. Benchmarks LEO

#1 Generalization

- What do we mean?
- How is it different than other ML methods?
- What can we do?

Generalization – What?

- Given a domain, we solve instances using GP
- The resulting model performs well over those instances
- Did we sample the space well? Will new data surprise us?
- Also, in practice, data ‘drifts’ and changes over time… will the GP model perform well?

Generalization – Different?

- Nope, not at all!
- All data driven methods suffer from this
- How do find a model that best represents the “true” underlying thing that generates the data?

- Unscientific evidence suggests a lot of GP papers do not adequately address this… why?

Generalization – Different?

- Asymmetrical data/algorithm in GP
- GP requires a lot of algorithm prep work, more than say a Random Forest or Neural Net
- Also, GP is often applied in data starved areas (robot control) or particularly challenging problems (nonlinear model)
- Does that effort, combined with higher computational cost, lead to lack of rigor with generalization?

Generalization – Different?

- Another difference…
- GP finds models and parameters that “can” be inspected
- Easier path to domain subject matter expert acceptance
- Easier transport of models to other environments… if (a big if) no tricks inside evaluation
- Compare to Random Forest, Neural Net, etc

Generalization - How

- Good problem setup
  - Train, test, validation and data processing
- Thoughtful objective function creation
  - No tricks or shortcuts, should be real-world like
- Good honest empirical evaluation
  - Significant, meaningful and SME validated
- After that, same as all other supervised ML
  - Put ‘control system’ around to monitor drift, etc.

Generalization – Next Steps

- More research into generalization significance, is asymmetric problem real?
- More research into solution simplification for SME validation and model transport
- More rigor in reviewing… one problem, one instance does make interesting results!
- What else?
- Questions?

Further Reading

- Kushchu, I. An evaluation of evolutionary generalization in genetic programming. Artificial Intelligence Review 18, 1 (2002), 3–14. Nice introduction, overview and example.
- Gagne, C., Schoenauer, M., Parizeau, M., and Tomassini, M. Genetic programming, validation sets, and parsimony pressure. In EuroGP2006. Using train,test,validation data plus complexity pressure was best.

#2 The Complexity of GP

- Not run time, although related
- Complexity of user experience and implementation
- How do we make it simpler out of the box?

Complexity – Run time

- Heuristic search method, slow
- Search over structure and parameters, slow
- Dependent on an evaluation function, slow
- Usually requires solution ‘compile’, slow

- In terms of algorithm complexity, no magic bullets, but many opportunities for improving
- … come back to this… but

Complexity – of GP

- How to make it simpler… what do we mean
- For example, some of the decision points
  - Functions, terminals
  - Objective function
  - Initialization
  - Operators
  - Selection pressure and operators
  - Bloat / size pressures
  - Stopping criterion
  - Oh, and generalization setup!

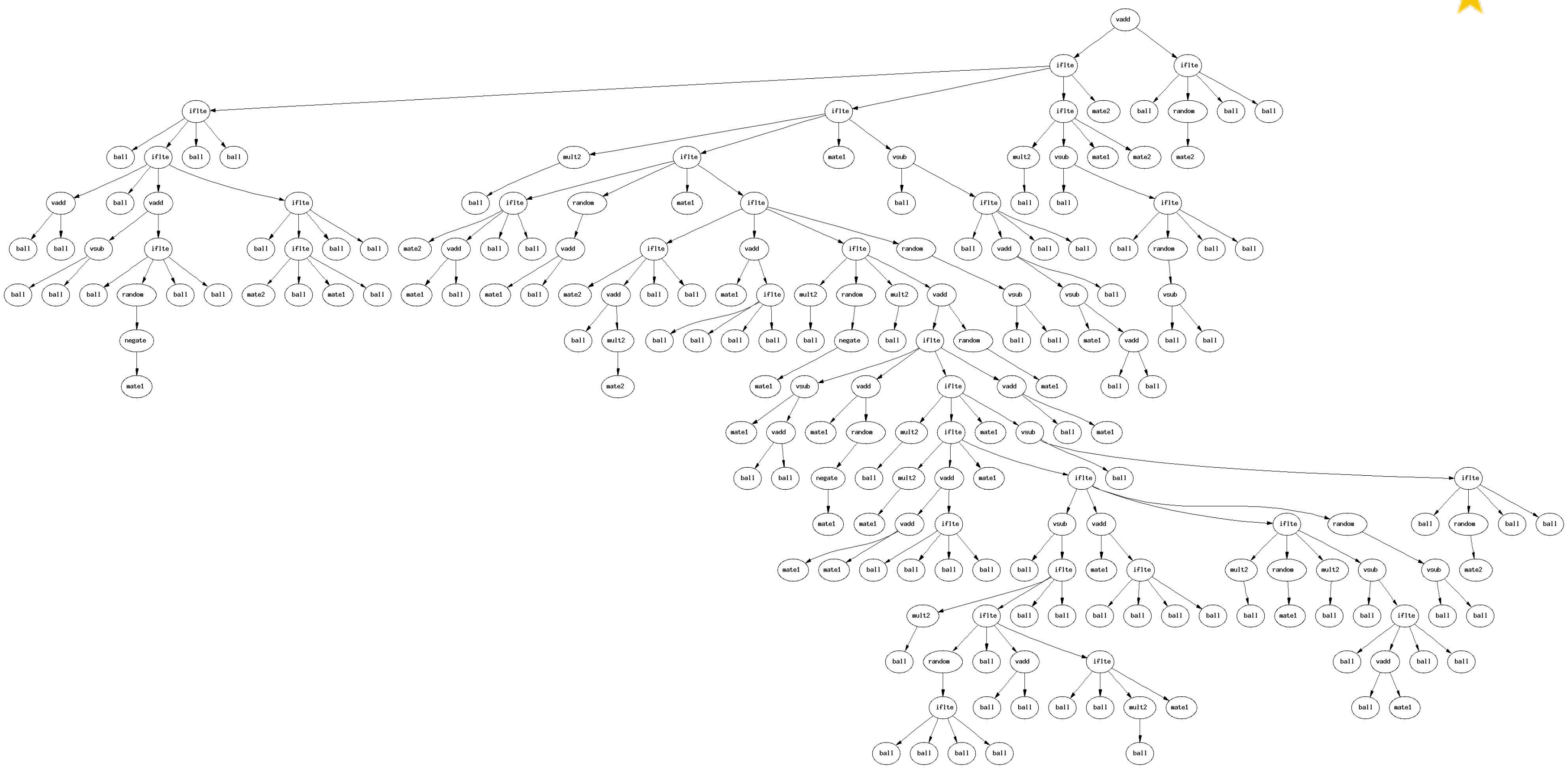
Complexity – of GP and more

- Within each decision, even more
  - Operators: interactions and mix
  - Selection pressure: replacement, archives, etc
- What is a minimal, viable GP algorithm?
  - Koza is still most often cited
  - Should it be by domain… GP-SR, GP-robot, etc
- Or, do we do meta-GP, add more algorithm complexity for simpler User Experience?

Complexity – One more thing

- Model interpretation is big benefit
- However, complexity of solutions high (big trees)
- Given a solution, which is product of evolution, i.e. non-”efficient” search, how to minimize with performance tradeoffs
- Should it be a post-search add-on or handled with Pareto archives, multi-objective, bloat / complexity control?

Complexity – One more thing



Complexity – Next Steps

- Research on simplification, approximations and Knowledge Extraction
- Research into Minimal, Viable GP (not smallest code base!)
- … and a canonical paper (not Koza '92)
- … that could be YOU!
- Research into VERY FAST systems… cloud, GPU, whatever… automating search can only help! (plug for MIT+GE work)

Further Reading

- Meta-learning – evolve an GP with a EA/GP, search parameter space
- Bloat, growth control – find least complex solutions
- Tiny GP competition and results
- GP-like alternatives: Hill-climbing like search (Poli and Langdon book, others), Incremental Program induction (Schmidhuber), others

#3 Representations

- **Identifying appropriate representations for GP**

  …..ideally based on some measure(s) of quality that capture the relationship between the fitness landscape and the search process.

Difficulty of Representation

- Hard to impossible to identify an optimal GP representation

- …but given a better understanding of the relationship between representation and search, differentiation between alternatives may be possible.

What’s a Representation?

❖ Representation = “genetic” encoding + “genetic” operators

Representation for GP

- Individual **is** *OR* **represents/encodes** a program

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char* argv){
        float x=0.0, y=0.0, z=0.0;
        x=atof(argv[1]);
        y=atof(argv[2]);
        z=atof(argv[3]);
        x = 2.0*sin(y) + 4.0*sin(x);
        z = (x*x) + exp(z);
        printf("The answer is: z=%f\n",z);
        return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void turnLeft(float degrees);
void turnRight(float degrees);
void moveForward(float distance);

int main(int argc, char* argv){
        turnLeft(90);
        if(sensorValue(0) > 1000)
                moveForward(10);
        else
                turnRight(90);
        return(0);
}
```

Attributes of GP

"Tell the computer what to do, not how to do it."

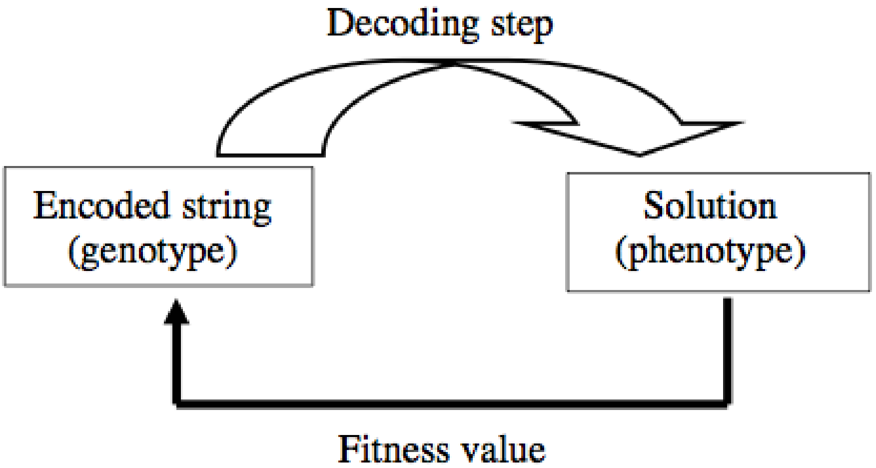
Arthur Samuel, 1959

John Koza's (1999) AP Attributes…

- Start with **high-level problem description** that results in a solution in the form of a computer program
- Automatically determine the programs **size and architecture**
- Automatically organise a group of **instructions** so that they may be **re-used** by a program
- **Problem-independence**
- **Scalability** to larger versions of the same problem
- Capability of producing **human competitive results**
- Evolutionary Automatic Programming/Genetic Programming…

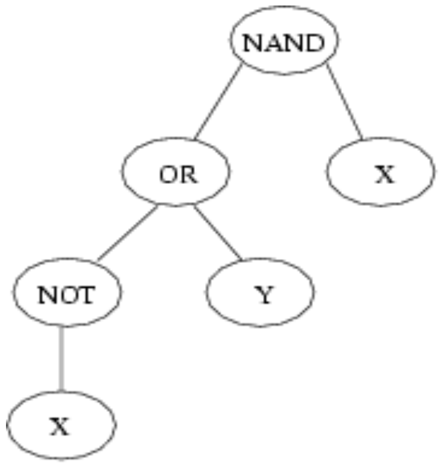
Many Representations for GP

- machine code, FSA, trees, strongly-typed, graph, linear, linear-graph, grammars, generative/developmental…



“Vanilla” GP

Koza popularised Lisp S-expressions…



(NAND (OR (NOT X) Y) X)

- Expressions (trees) generated from
  - Function Set: boolean, arithmetic, loops, user-defined functions…
  - Terminal Set: inputs, constants, variables, …

“Vanilla” GP Features

Sufficiency

- Function+Terminal sets: powerful enough to represent a solution

Parsimony

- Smaller Function+Terminal sets are better

Closure

- Each function should gracefully handle all values it ever receives
- (/ 5 0) !?!

Evolving Representation!

❖ Why not "evolve it"!?!?

❖ Examples include…
- Langdon's Data Structures
- Spector's "Autoconstructive" Evolution
- Banzhaf's Evolution of "genetic code"
- ?Hyperheuristics

Representation Open Issues

- What is the best representation for my problem?
  - computationally "sufficient"
  - facilitate navigation (e.g., see Semantic Operators later)
  - automatically identify and manipulate abstractions/modules
  - handle variable dimensions…
- How do we compare representations?
  - E.g., locality, redundancy and scaling
- Need more rigorous/formal analysis…

Representation Open Issues

- What ever happened to evolving algorithms?

#4 Modularity & Scalable GP

- Define a clear measure of success for what it means to achieve "Scalable GP" as well as modularity
- How well does GP scale to problems of increasing complexity/difficulty?
- How can we improve scalability of GP?
- Given representations in GP can evolve, what is scalability in GP anyway?

Approaches to Modularity

- Many approaches…
  - E.g., ADF's, architecture-altering operators (Koza), Genetic Library Builder (GLiB - Angeline & Pollack), Adaptive Representation through Learning (Rosca), Automatically defined macros (Spector), reuse of "concepts" (Seront), lambda abstraction (Yu), linear-gp register reuse & repeated patterns (Langdon & Banzhaf), module repository (Majeed & Ryan), sub-graph encapsulation (Walker & Miller), Run Transferable Libraries (Keijzer et al), functional modularity (Krawiec & Wieloch), grammar-defined functions (O'Neill, Hemberg, Harper, Swafford), Swafford PhD thesis (2013)

Modularity & Evolution

- Modularity & Evolvability (Altenberg)…modularity may have positive effect on “alignment” between spaces of phenotypic variation and selection gradients

- Evidence for dynamic environments leading to emergent modularity (Kashtan)

Modularity

- Mechanism to
  - protect parts of individuals from disruption
  - achieve abstraction and parameterisation

- What's the best way to
  - automatically **identify** modules?
  - achieve automatic **abstraction**
  - achieve automatic **manipulation** of modules
    - E.g., architecture altering operators

- Is modularity critical to scalability?
- How are modules used? Can their use guide search operators?

- Can we achieve modularity, hierarchy and reuse in a more principled manner? (e.g., software eng – or should we even consider these human-centric approaches?)

#5 Dynamic Problems

- The "natural environment" for artificial evolution.
- Dynamic in so many ways:
  - Type of change (e.g., constraints, fitness landscape, combinations)
  - Degree
  - Frequency
  - Combinations of all of the above!
- Mind-shift from optimisation to "survival"

Dynamic Problems with GP

- Can borrow strategies from broader EC literature
  - E.g., Branke (2001) and Morrison (2004)

- GP inherently dynamic!
  - Co-evolving
    - Structure
    - Parameters
  - Dimensionally dynamic!

Dynamic Problems – lots to do!

- Little formal analysis of GP in dynamic env's
  - E.g., Bloat - Langdon & Poli (1998), population sizing - Tu & Banzhaf (2009) & Vanneschi (2009), constant evolution – time series - Dempsey (2009)

- Recall (Kasthan) emergent modularity
  - Also, dynamic environments can provide more efficient search
  - some evidence for GP in specific cases (O'Neill)

#6 Open-ended Evolution

- Designing an evolutionary system capable of continuously adapting and searching…(can also mean un-directed search)

- Essential
  - Feedback loops
    - Dynamic environment
    - Co-evolutionary processes
  - Continuously injected randomness

#6 Open-ended Evolution

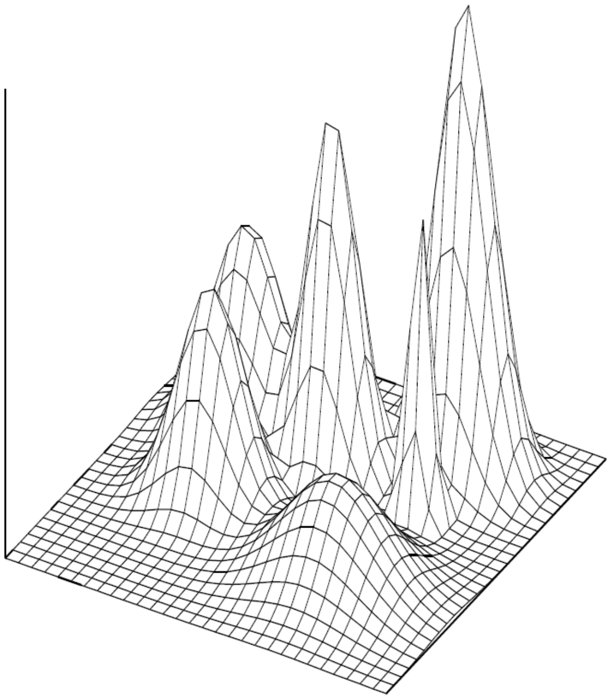
- EC & ALife have failed!
  - What are the missing ingredients for artificial evolution to achieve the open-ended emergence of complexity, innovation and adaptation witnessed in nature?

#7 Fitness Landscapes and Problem Difficulty in GP

Fitness Landscape

Fitness landscape $(\mathcal{S}, \mathcal{V}, f)$:

- $\mathcal{S}$: set of potential solutions,
- $\mathcal{V} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$: neighborhood function,
- $f : \mathcal{S} \rightarrow \mathbb{R}$: fitness function.



$\mathcal{V} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$: neighborhood function

$\forall x \in \mathcal{S}$,

$$\mathcal{V}(x) = \{y \in \mathcal{S} \mid y = op(x)\}$$

$$\mathcal{V}(x) = \{y \in \mathcal{S} \mid d(y, x) \leq 1\}$$

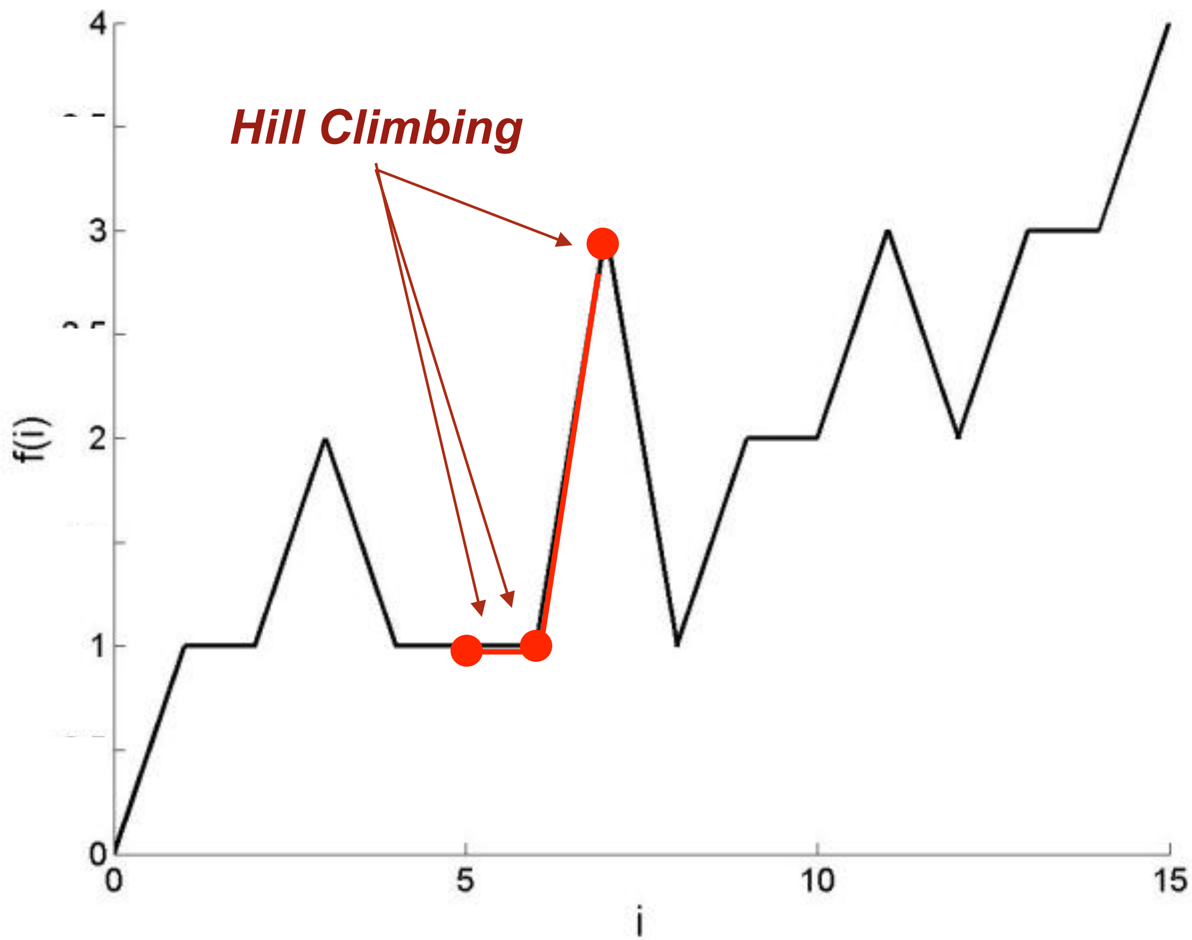
Example

$S = \{ i \mid \quad i \in \mathbb{N} \quad \& \quad 0 \leq i \leq 15 \}$

$\forall i \in S, \quad f(i)$ = number of "1"s in the binary representation of i

Neighborhood: $\quad j \in N_i \Leftrightarrow |j - i| = 1$

Fitness Landscape



Remark that...

if we consider exactly the same problem, but with a ***different neighborhood structure***:

$S = \{ i \mid \quad i \in \mathbb{N} \quad \& \quad 0 \leq i \leq 15 \}$

$\forall i \in S, \quad f(i)$ = number of "1"s in the binary representation of i

Neighborhood: $j \in N_i \Leftrightarrow j$ and i differ by just 1 bit

There are ***no local optima*** in this fitness landscape!
(every individual that is different from the global optimum has at least one neighbor better than him, that can be obtained by changing a 0 into a 1).

Another Case ("CONO" Problem)

S = { vectors of prefixed length of real numbers included in [0,10] }

$\forall\, i \in S$, $f(i)$ = distance to a prefixed (and known and unique) global optimum

Neighborhood: $j \in N_i \Leftrightarrow$ *j is equal to i except for the random perturbation of some of its coordinates of a quantity included in [0,1].*

Example

The global optimum → [8.0, 6.0, 4.0, 7.0, 5.0]

A solution i → [5.2, 6.4, 2.1, 4.9, 3.7]

A solution j neighbor of i → [5.8, 6.4, 2.9, 4.9, 3.6]

closer!

Unimodal Fitness Landscape

Importance of Fitness Landscape

It gives a *visual intuition* of the *facility* or *difficulty* of a search agent (like Hill Climbing, but also Evolutionary Algorithms) ***to find the global optimum***.

- Smooth landscape, with only one "peak" (global optimum)
  easy problem
- Rugged landscape, with many local optima
  hard problem

Limitation of fitness landscapes

It is generally ***impossible to draw*** a fitness landscape:

- Huge search space
- Huge neighborhoods *(multi-dimensionality!)*

Objective of Problem Hardness Studies

Find indicators of problem hardness (that typically capture some important characteristics of the fitness landscape and) that can give insight on the ability of a GP configuration to solve the problem....

- Without having to run GP
- Without having to draw the fitness landscape

Autocorrelation [Kinnear, 1994]

Proposed measure of problem hardness for GP: ***autocorrelation function*** (Weinberg in 1990 and Manderick in 1991 had studied the same measure for GAs).

Autocorrelation of $(f(s_1), f(s_2), \ldots)$ along a random walk $(s_1, s_2, \ldots)$ (Weinberger 1990 [29]) :

$$\rho(n) = \frac{E[(f(s_i) - \bar{f})(f(s_{i+n}) - \bar{f})]}{var(f(s_i))}$$

autocorrelation length $\tau = \frac{1}{\rho(1)}$

- small τ : rugged landscape
- long τ : smooth landscape

Basically no clear relationship between autocorrelation values and problem hardness was observed for GP

Difficulty and Neutrality [T. Yu, J. Miller 2001]

Larger amount of neutrality allow GP to generate fitter individuals, in particular for hard problems

(results criticized in [Collins, 2005])

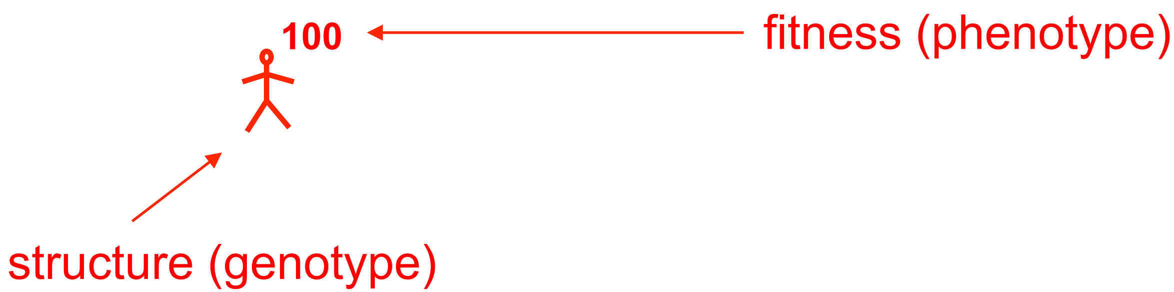
Fitness-Distance Correlation (fdc)

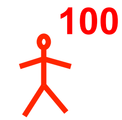
(Introduced in [Jones et al., 1995] for GAs)

Hypothesis: what makes a problem *easy* or *hard* is the ***relationship*** between ***fitness*** of individuals and the ***structural similarity*** of individuals to the optimum.

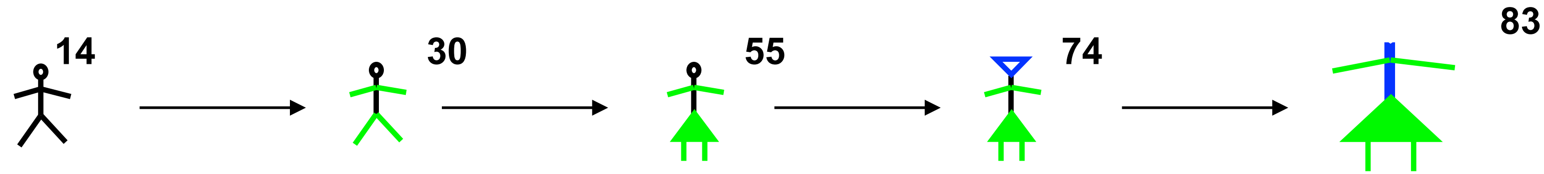
Example

Let's suppose that the global optimum is:



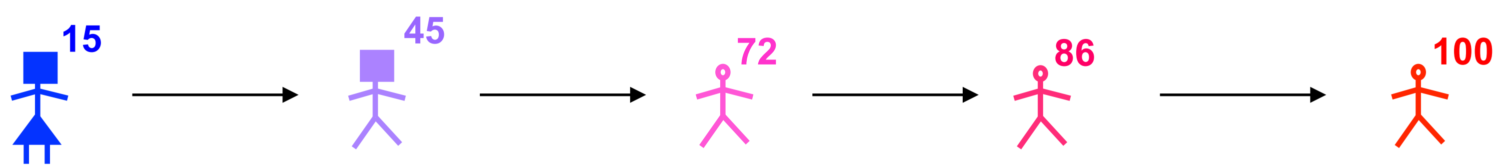
optimum = 

Difficult Problem



the more fitness increases (improves), the more individuals are <u>different</u> from the optimum

Easy Problems



the more fitness increases (improves), the more individuals are <u>similar</u> to the optimum

Fitness Distance Correlation (*fdc*) *[T. Jones, 1995]*

Given a sample of *n* individuals, let's suppose to know:

- the set $F = \{f_1, f_2, \ldots, f_n\}$ of the individual fitnesses
- the genotype of the global optimum (individual with the best fitness)
- a measure to express the genotypic distance between individuals

Let $D = \{d_1, d_2, \ldots, d_n\}$ be the *n* distances to the global optimum, then

fdc* is the correlation between sets *F* and *D

Summary of the Results obtained by *fdc* in GP [Vanneschi et al., 2004]

Fdc is a ***very reliable*** measure of difficulty for a large set of problems.

Fdc also has some drawbacks:

- Counterexamples exist
- It is not predictive

  Optima must be known « a priori »
  (this makes fdc « almost» unusable in practice)

A new measure must be defkned to quantify the difficulty of real-life problems.

A measure that has been proposed is based on the concept of ***fitness cloud.***

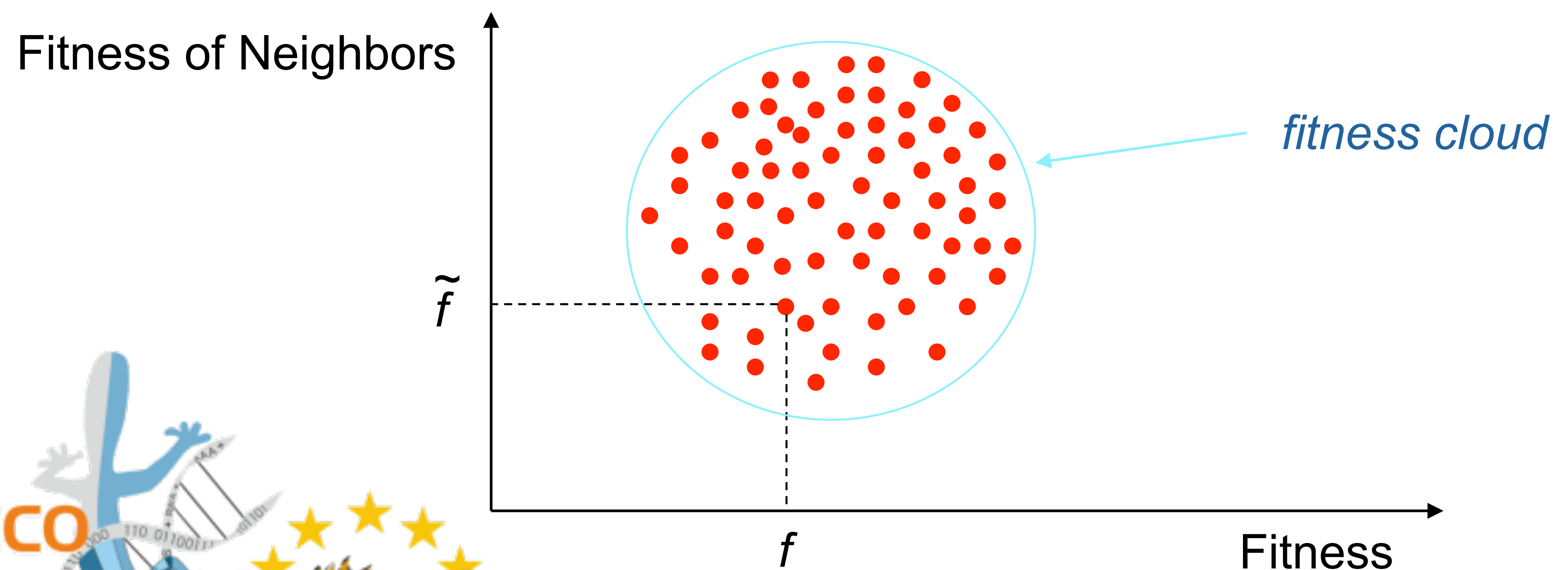
Fitness Clouds

(First introduced for binary landscapes in [Vérel et al., 2003])

For each individual γ (in a sample or in the whole search space) a point is plotted:

- **abscissa** = fitness value of γ
- **ordinate** = fitness value of a "particular" neighbor (chosen randomly or by some particular techniques).

here: *neighbor = individual obtained by applying <u>one step of mutation</u> to* γ



Negative Slope Coefficient (nsc) [Vanneschi et al., 2004]
Summary of results

- Good hardness indicator for:
  - Trap Functions
  - Royal Trees
  - Binomial-3 Problem [Daida *et al.*, 2001]
  - Even Parity Problem [Koza, 1992]
  - Artificial Ant on the Santa Fe Trail [Koza, 1992]
  - ... (also some real-life applications)
- Many ways of calculating the *nsc* have been used:
  - Number of neighbors for each sampled individual
  - Number of mutations to generate neighbors
  - Different types of mutations to generate neighbors
  - Different techniques to partition the fitness clouds into bins

- *nsc* is predictive $\Rightarrow$ it can be used on *any* problem
- *nsc* has not been normalized yet into a given range (classification of different problems by their difficulty)
- A theoretical justification for nsc in [Poli, Vanneschi, GECCO 2007]. Some problems of the nsc for GAs in [Vanneschi, Valsecchi, Poli, GECCO 2009]

#8 Semantics in GP

GP as a Machine Learning Method (supervised learning)

- **Known**: the correct outputs for a fixed given set of inputs $\{I_i, O_i\}$
- **Sought**: a function belonging to a certain class that interpolates those points, i.e., $f(I_i) = O_i$ for any i
- **Output vector**: the vector of the outputs of f is $f(I)=(f(I_i))$
- **Fitness** on the error on the training set, i.e.,***distance*** betw tors of f and the target output vector $F(f)=D$ (ERROR AS DISTANCE)

semantics ([Moraglio et al., 2012] and many others...)

Semantic Diversity #1

[McPhee et al., 2008]

Use of truth tables to analyze behavioral changes in crossover for boolean problems

Considered the semantics of two components in each tree: semantics of **subtrees** and semantics of **context** (the remainder of an individual after removing a subtree).

Measured the **variation** of these semantic components throughout the GP evolutionary process.

Fixed-semantic subtrees: subtrees such that the semantics of the entire tree does not change when they are replaced by another subtree.

There may be **many fixed semantic subtrees** ***when the tree size increases*** during evolution; thus it becomes very difficult to change the semantics of trees with crossover and mutation.

Semantic Diversity #2

[Beadle and Johnson, 2008]

Semantic is used to define an algorithm called **Semantically Driven Crossover** (**SDC**).

SDC has been developed based on analysis of the behavioral changes caused by crossover.

Use of a canonical representation of members of the population (Reduced Ordered Binary Decision Diagrams-ROBDDs) to check for semantic equivalence without having to access the fitness function: two trees are semantically equivalent if and only if they reduce to the same ROBDD.

This is used to determine which participating individuals are copied into the next generation.

If the offspring are semantically equivalent to their parents, the children are discarded and the crossover is repeated.

Increased semantic diversity in the evolving population, and a consequent improvement in the GP performance.

Semantic Diversity #3

[Beadle and Johnson, 2009]

Previous work extended to mutation: semantics is used to test the effects of behavioral control at the point of the mutation operator.

Presented **Semantically Driven Mutation** (**SDM**), which can explicitly detect and apply behavioral changes caused by the syntactic modifications in programs caused by mutation.

SDM does not allow mutated programs to be produced when they are behaviorally equivalent to the original program. The aim of this is to avoid getting stuck in areas of the semantic/search space that have already been investigated.

As in [Beadle and Johnson, 2008], the key feature of the semantically driven operator is the ability to canonically represent programs in such a way that it is possible to compare them, looking for equivalent behaviors.

Semantic Locality

[Nguyen et al., 2010]

Investigation of the role of **syntactic** locality and **semantic** locality of **crossover**.

The results show that improving syntactic locality reduces code growth, and that leads to a slight improvement of the ability to generalize.

By comparison, improving semantic locality significantly enhances GP performance, reduces code growth and substantially improves the ability of GP to generalize.

Semantic Diversity + Locality #1

[Nguyen et al, 2009(a)]

Semantics Aware Crossover (**SAC**), a crossover operator promoting semantic diversity, based on checking semantic equivalence of subtrees.

[Nguyen et al, 2011]

Extended to **Semantic Similarity based Crossover** (**SSC**), which turned out to perform better than both standard crossover and SAC

Objective: *incorporate semantics* into the design of new crossover operators, so as to maintain greater semantic diversity and provide higher locality than standard crossover.

Semantic Diversity + Locality #2

[Nguyen et al, 2009(b)]

SSC extended to mutation leading to **Semantic Similarity based Mutation** (**SSM**).

Superior performance of SSM compared to standard mutation.

Geometry in the Semantic Space

[Krawiec and Lichocki, 2009] + [Krawiec, 2012]

Proposed a class of crossover operators for GP aimed at making offspring programs intermediate (or medial) with respect to parent programs in the semantic space (**geometric**).

Suggested that the prospects of designing a crossover operator that works in the genotype space and behaves geometrically in the corresponding semantic space are gloomy in GP, given the *complexity of the genotype-phenotype mapping*.

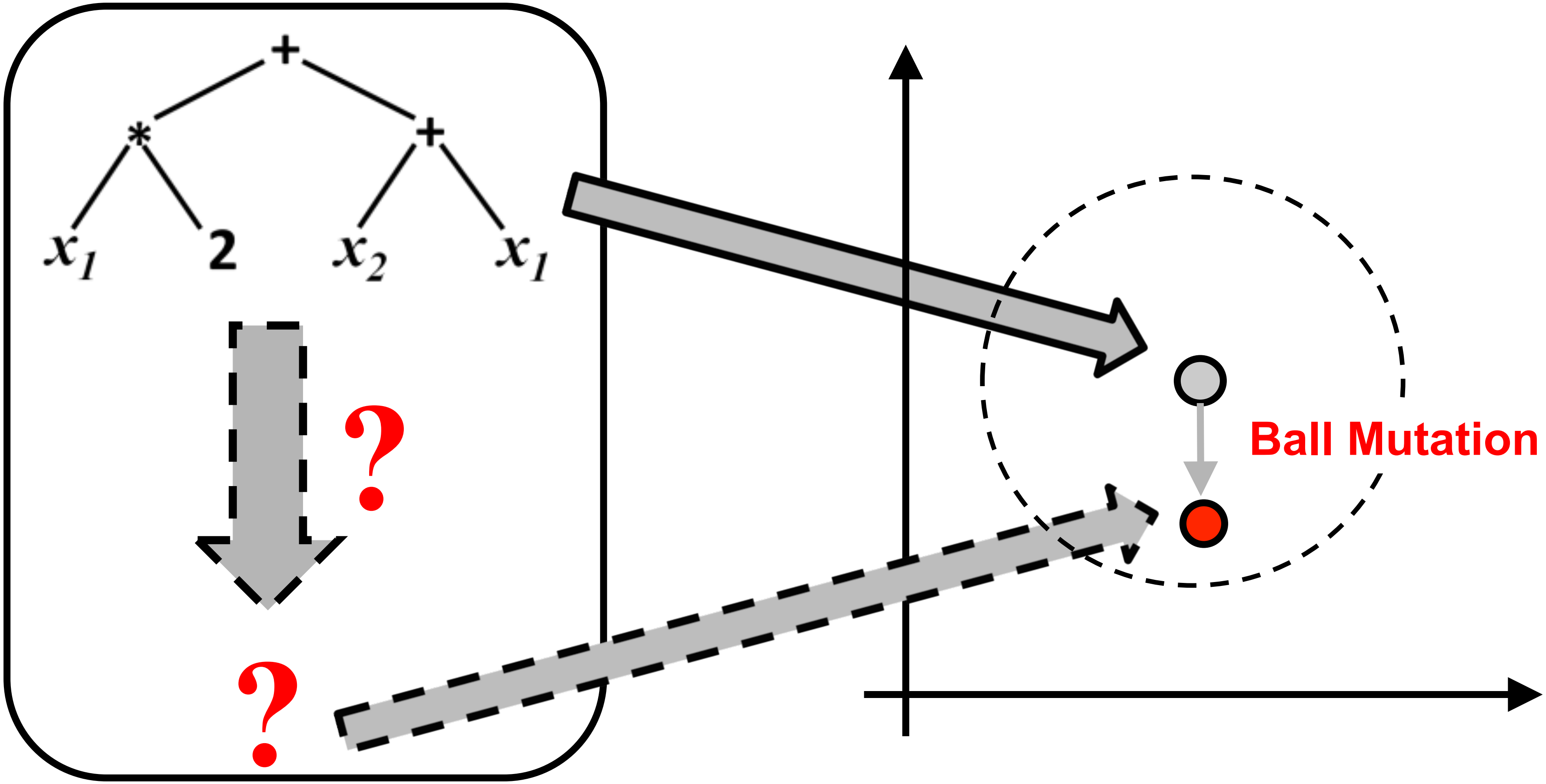
Hence, rather than guaranteeing the geometric behavior, their operator tries to approximate it by analyzing the offspring after it has been created.

This limitation is overcome by the geometric semantic operators proposed in [Moraglio et al., 2012], discussed in the continuation.

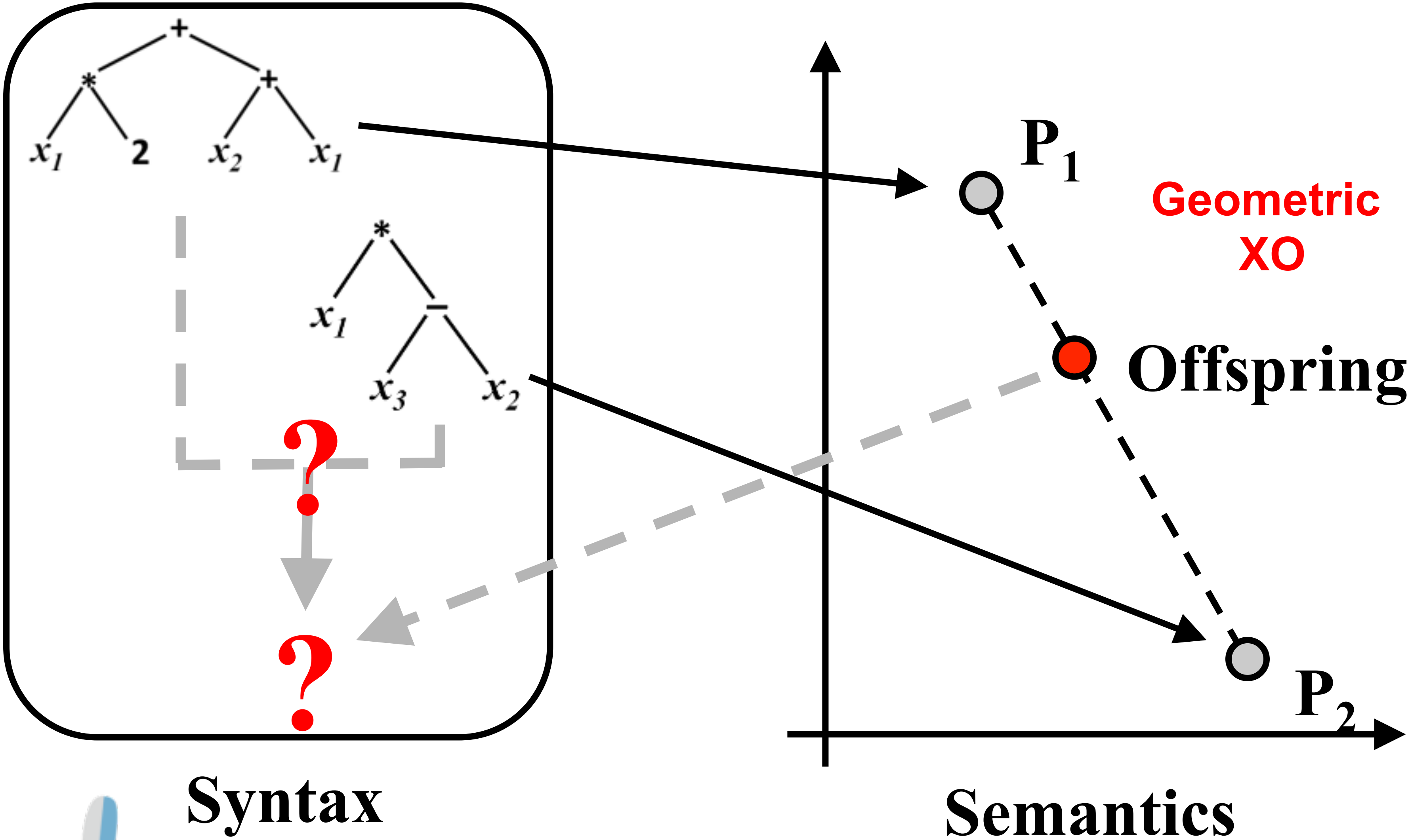
GP Geometric Semantic Operators
[Moraglio et al., 2012]

Objective:

Is it possible to define transformations on the syntax of individuals that have **known effects** on their semantics?



Syntax **Semantics**



+
*
+
x1
2
x2
x1
*
x1
–
x3
x2
?
?
P1
Geometric XO
Offspring
P2
Syntax
Semantics

Ball Mutation on the Semantic Space

If
a transformation on the syntax of an individual whose effect is ***ball mutation on the semantic*** space can be found

Then

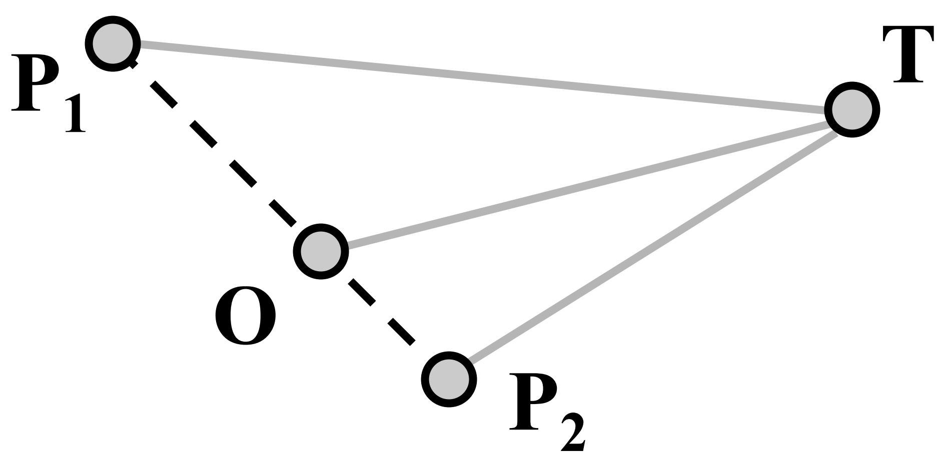
a **unimodal fitness landscape** can be induced on ***any*** problem consisting in matching input data into known targets (e.g. regressions and classifications)

Problem mapped into the "CONO".

Geometric XO on the Semantic Space

If
a transformation on the syntax of individuals whose effect is ***geometric crossover on the semantic*** space can be found

Then
the offspring is **not worse than the worst of its parents**



Is it a dream?

Yes... but turning into reality

Those operators have been defined:

A. Moraglio, K. Krawiec, and C. G. Johnson.
Geometric semantic genetic programming.
In C. A. Coello Coello, et al., editors, *Parallel Problem Solving from Nature, PPSN XII* (part 1), volume 7491 of Lecture Notes in Computer Science, pages 21–31. Springer, 2012.

Geometric Semantic Crossover [Moraglio et al., 2012]

Definition 1. (Geometric Semantic Crossover). *Given two parent functions* $T_1, T_2 : \mathbb{R}^n \rightarrow \mathbb{R}$*, the geometric semantic crossover returns the real function* $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$*, where* T_R *is a random real function whose output values range in the interval* $[0, 1]$.

T_{XO} =

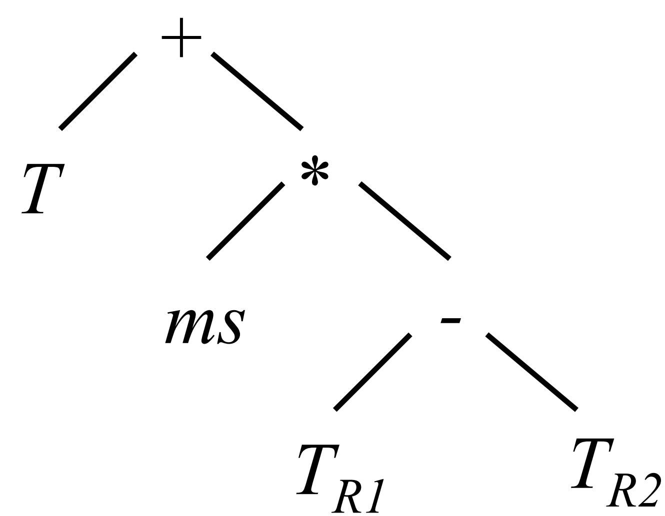
```
              +
            /   \
          *       *
         / \     / \
       T1   TR  -    T2
               / \
              1   TR
```

T_R = Random function with codomain [0, 1]

Geometric Semantic Mutation [Moraglio et al., 2012]

Definition 2. (Geometric Semantic Mutation). *Given a parent function* $T : \mathbb{R}^n \to \mathbb{R}$*, the geometric semantic mutation with mutation step* ms *returns the real function* $T_M = T + ms \cdot (T_{R1} - T_{R2})$*, where* T_{R1} *and* T_{R2} *are random real functions.*

T_M =



T_{R1} , T_{R2} = Random functions

Drawback of Geometric Semantic Operators

These operators, by construction, always produce offspring that are larger than their parents, causing an ***exponential*** growth in the size of the individuals [Moraglio et al., 2012]

This renders them ***useless in practice***.

A solution that has been proposed: "***simplification***" of the individuals during the evolution. But….

An Efficient Implementation

In:
A New Implementation of Geometric Semantic GP Applied to Predicting Pharmacokinetic Parameters.
L. Vanneschi, M. Castelli, L. Manzoni, S. Silva.
Accepted for publication in the *EuroGP 2013* Proceedings
Lecture Notes in Computer Science.

We propose a ***new implementation*** of Moraglio's geometric semantic operators that is ***efficient*** and does not imply any online simplification phase and thus allows us to use them on complex real-life applications!

Summary of the results obtained

- An ***efficient implementation*** of geometric semantic operators, that has allowed us to use them on real-life applications.
- ***Excellent results*** on the studied ***applications***.
- ***New insights about the generalization*** ability of geometric semantic operators (without the novel implementation that allowed us to use geometric semantic GP on these complex real-life problems, this interesting property would probably remain unnoticed).

Major open issue

The ***reconstruction of the expression of the best individual***, even though we do it only once and after the termination of the run, is still an issue:

Individuals after hundreds of generations get so huge that it may be *impossible* to reconstruct their entire expression (even though it is possible to get some information about it, such as the features or primitives it uses...).

Models generated by geometric semantic GP are ***black*** (or at least ***"dark gray"***) boxes!

We are working on this!

GECCO 2013

#9 The Influence of Biology

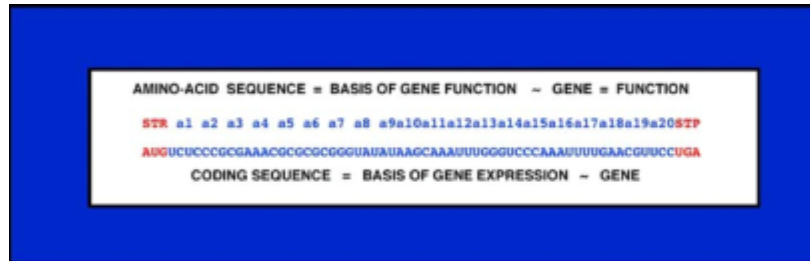
- Weaknesses of current paradigm
- The trade-off
- Opportunities
- Reversing the flow

Weaknesses of current Paradigm

- Fixed representation
- Static fitness functions
- Closed systems
- Our notion of genes
- Simple maps from genotype to phenotype
- Pre-determined operator features
- No role for non-expressed material
- Direct passing of genes without further qualifications
- Scalability

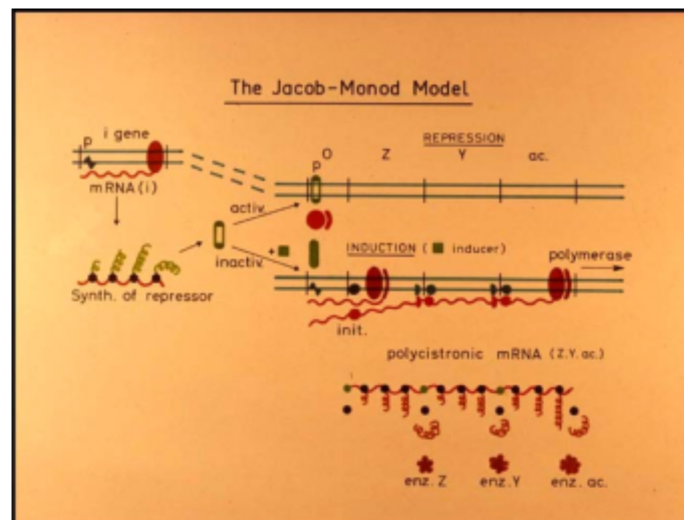
Our Notion of Genes …

- From …
- EC genes that fully determine phenotypic outcomes
- Genes as "coding sequences"



From: Scherrer/Jost: Theory Biosci., 2007

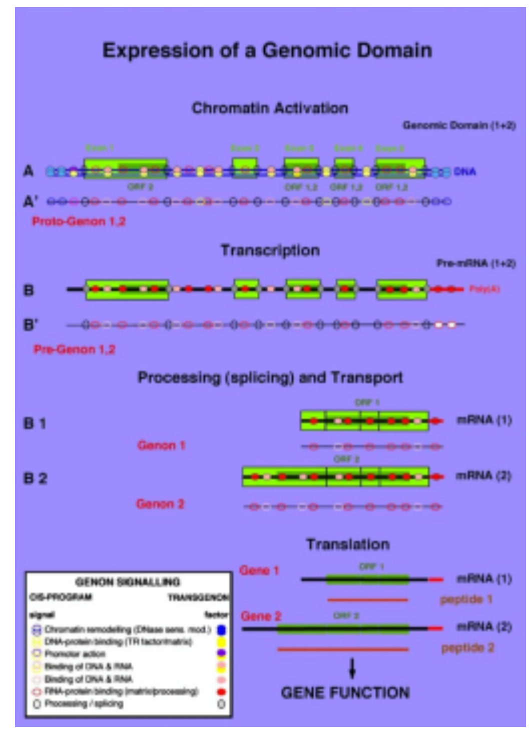
- … to …
- Genes as regulating units
- The operon model



… has to develop …

- … to
- Expression management of
- Highly intricate complexes

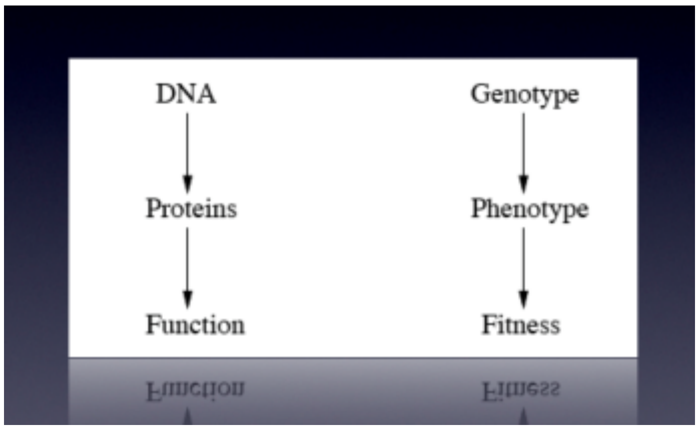
- Chromosomes and chromatin structures
- Regulation, transcription, splicing
- Editing of intermediate products (RNA)
- Translation in ribosomes
- …. to function



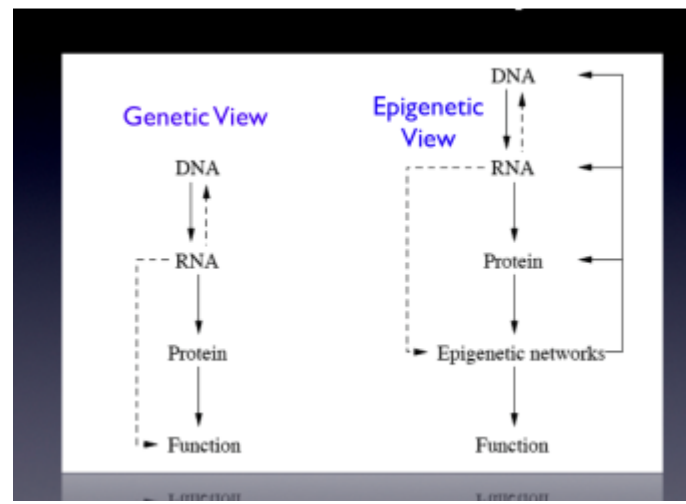
From: Scherrer/Jost: Theory Biosci., 2007

The Central Dogma is dead

- The linear flow of information from DNA
- to function is a grave oversimplification



- The picture now is complex and bidirectional,
- closing loops and forming networks



From: Banzhaf et al: Nature Rev. Gen., 2006

No role for non-expressed material

Organism	No. of protein coding genes	Genome Size (Mb)	Coding sequences (Mb)	Coding sequences (%)	Transcribed non-coding sequences (Mb)	Transcribed non-coding sequences (%)	Ratio
Human	20-25,000	2851	34	1.2	1619	57	48:1
Mouse	20-25,000	2490	31	1.3	1339	54	43:1
Fruit Fly	~13,500	120	22	18	53	44	2.4:1
Nematode	~19,000	100	26	26	33	33	1.3:1

From: Frith,Pheasant, Mattic: Eur. J. Hum., 2005

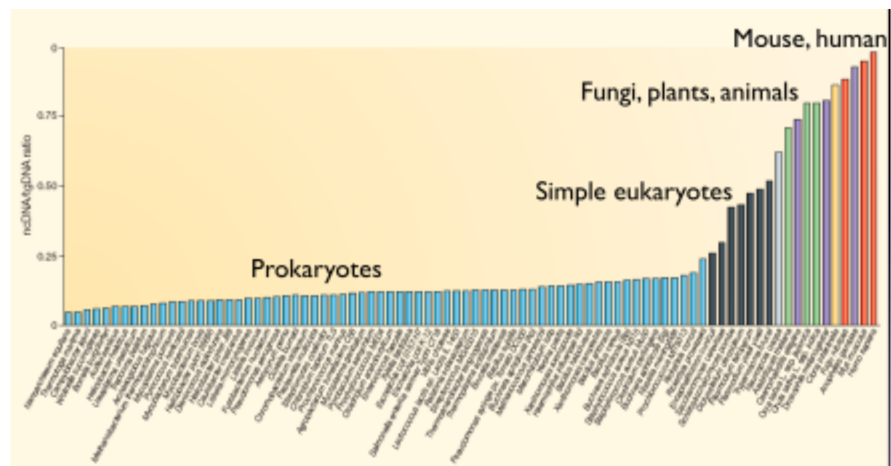


Figure 1 | **The ratio of non-coding to protein-coding DNA rises as a function of developmental complexity.** Prokaryotes have less than 25% non-coding DNA, simple eukaryotes have between 25 and 50% non-coding DNA and more complex fungi, plants and animals have more than 50%, rising to approximately 98.5% non-coding DNA in humans — which also have a genome size that is three orders of magnitude larger than prokaryotes. Note that this analysis corrects for ploidy, whereas pre-genomic estimations of the amount of DNA in different organisms did not. The different colours represent prokaryotes (bacteria and archaea) (blue), simple eucharyotes (black), *Neurospora crassa* (grey), plants (green), non-chordate invertebrates (nematodes, insects) (purple), *Ciona intestinalis* (urochordate) (yellow) and vertebrates (red). ncDNA, non-coding DNA; tgDNA, total genomic DNA. Reproduced with permission from REF. 77 © (2003) BioMed Central Ltd.

Tafts, Mattick, Increasing biological complexity is positively correlated with the relative genome-wide expansion of non-protein-coding DNA sequences, <www.arxiv.org/abs/q-bio.GN/0401020>, 2003

Trade-Offs

- Level of abstraction in models
- Potential for harvesting useful features
- Increased simulation time for evolutionary processes
- More emergent phenomena ?

Opportunities

- Epigenetics
- Multi-level selection
- Regulatory networks
- Multi-cellularity and Development
- Self-modifying genomes
- Research into novelty, innovation and creativity

#10 GP Needs Benchmarks

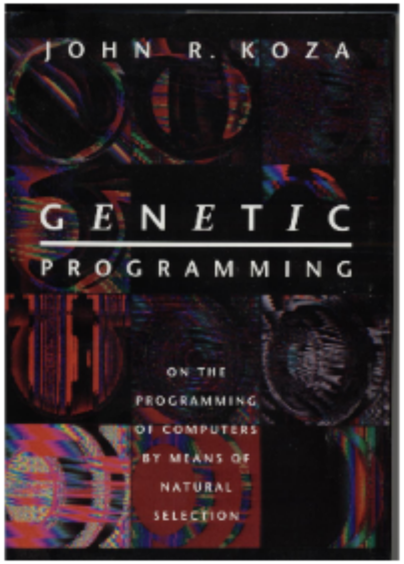
Published Use of Benchmarks

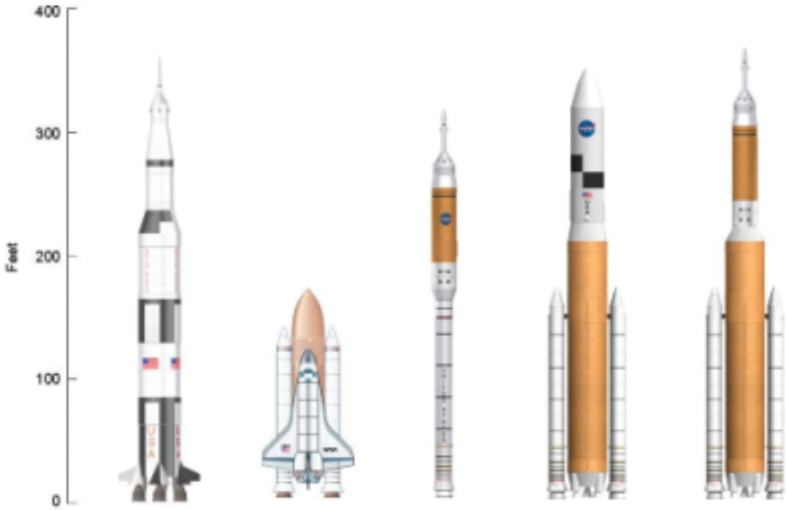
Survey of EuroGP and GECCO's GP Track from 2009 to 2011.
183 articles using 471 problem instances.

	Percentage (nearest percent)
Symbolic Regression	32
Classification	27
Path Finding and Planning	10
Boolean Functions	9
Traditional Programming	8
Predictive Modelling	7
Constructed Problems	3
Control Problems	1
Others	4

Limited variety e.g. 26% of papers involving symbolic regression used the quartic equation.

“De Facto” Banchmarks





What makes for a good benchmark?

- Tunably Dicult
- Varied
- Relevant (Real World? Constructed?)
- Fast (?)
- Accommodating to Implementors
- Supports good empirical method (e.g. problem generation)
- Easy to interpret and compare
- Representation Independent
- Precisely Defined (to an extent!)
- Known global optimum?

A Good Starting Point

GP Benchmarks.org

James McDermott
David R. White
Sean Luke
Luca Manzoni
Mauro Castelli
Leonardo Vanneschi
Wojciech Jaskowski
Krzysztof Krawiec
Robin Harper
Kenneth De Jong
Una-May O'Reilly
... and many many others (sorry if I forgot to include your name!)

#11 Miscellany…

- Algorithm Induction
- Halting Problem
- Domain Knowledge
- Usability
- GP Theory
- Constants
- Visualisation…

(…and we never mentioned BLOAT! ☺)

Conclusions

So many open issues...

- Can we increment GP generalization ability?
- Is there a better way to deal with programs' complexity?
- How to choose the right representation for a problem?
- What is the best way of using GP in dynamic environments?
- How can we measure/predict the ability of GP to solve a problem?
- How can we use sematic awareness to improve GP?
- Can we exploit the richness of nature better then we currently do?
- ...

One big objective...

Let GP become a trusted mainstream member of the computational problem solving toolkit.

Why not yet?

Questions & Discussion

Acknowledgements

- The impetus for this article arose out of the EuroGP 2008 debate on Grand Challenges of Genetic Programming which took place on 27 March 2008 at the Evo* event in Naples, Italy. In particular we thank the two other panel members, Nic McPhee and Riccardo Poli, and also the many members of the audience who participated in the debate. Many of these issues have been raised on multiple occasions at previous (and subsequent) EuroGP debates so this inspired us to put these ideas on paper to open the debate to a wider audience. MO’N acknowledges support of Science Foundation Ireland under Grant No. 08/IN.1/I1868 and 08/SRC/FM1389. WB acknowledges support from the Canadian National Science and Engineering Research Council (NSERC) under discovery grant RGPIN 283304-07.

References

- Altenberg, L. NK fitness landscapes. In Section B2.7.2 in Handbook of Evolutionary Computation (1997), T. Back et al., Ed., IOP Publishing Ltd and Oxford University Press, pp. B2.7:5 – B2.7:10.
- Altenberg, L. Modularity in evolution: Some low-level questions. In Modularity: Un- derstanding the Development and Evolution of Complex Natural Systems, D. Rasskin- Gutman and W. Callebaut, Eds. MIT Press, Cambridge, MA, USA, 2004. In press.
- Angeline, P. J. Two self-adaptive crossover operators for genetic programming. In Advances in Genetic Programming 2, P. J. Angeline and K. E. Kinnear, Jr., Eds. MIT Press, Cambridge, MA, USA, 1996, ch. 5, pp. 89–110.
- Archetti, F., Lanzeni, S., Messina, E., and Vanneschi, L. Genetic programming for computational pharmacokinetics in drug discovery and development. Genetic Pro- gramming and Evolvable Machines 8, 4 (Dec. 2007), 413–432. special issue on medical applications of Genetic and Evolutionary Computation.
- Asuncion, A., and Newman, D. UCI machine learning repository, 2007.
- Banzhaf, W. Editorial introduction to the first issue. Genetic Programming and Evolvable Machines 1 (2000), 5 – 6.
- Banzhaf, W., Beslon, G., Christensen, S., Foster, J., K´ep`es, F., Lefort, V., Miller, J., Radman, M., and Ramsden, J. From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics 7, 9 (2006), 729–735.
- Banzhaf, W., Francone, F. D., and Nordin, P. The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In 4th Int. Conf. on Parallel Problem Solving from Nature (PPSN96) (1996), W. Ebeling et al, Ed., Springer, Berlin, pp. 300–309.
- Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, San Francisco, CA, USA, Jan. 1998.
- Banzhaf, W., Poli, R., Schoenauer, M., and Fogarty, T., Eds. Proceedings of Ge- netic Programming, First European Workshop, EuroGP'98, Paris,France, April 14-15, 1998 (Berlin, 1998), vol. 1391 of LNCS, Springer.
- Beadle, L., and Johnson, C. Semantically driven crossover in genetic programming. In Proceedings of the IEEE World Congress on Computational Intelligence (Hong Kong, 1-6 June 2008), J. Wang, Ed., IEEE Computational Intelligence Society, IEEE Press, pp. 111–116.

References (contd.)

- Bhattacharyya, S., Pictet, O., and Zumbach, G. Representational semantics for genetic programming based learning in high-frequency financial data. In Genetic Pro- gramming 1998: Proceedings of the Third Annual Conference (University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998), J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, Eds., Morgan Kaufmann, pp. 11–16.
- Bianco, S., Gasparini, F., Schettini, R., and Vanneschi, L. An evolutionary frame- work for colorimetric characterization of scanners. In International Workshop on Evolutionary Computation in Image Analysis and Signal Processing, EvoIASP 2008. Proceedings of Applications of Evolutionary Computing, EvoWorkshops 2008 (2008), M. Giacobini et al., Ed., vol. 4974/2008 of Lecture Notes in Computer Science, LNCS, Springer, Berlin, Heidelberg, New York, pp. 245–254.
- Brameier, M., and Banzhaf, W. Linear Genetic Programming. No. XVI in Genetic and Evolutionary Computation. Springer, 2007.
- Branke, J. Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, 2001.
- Burke, E. K., Hyde, M. R., and Kendall, G. Evolving bin packing heuristics with genetic programming. In Parallel Problem Solving from Nature - PPSN IX (Reykjavik, Iceland, 9-13 Sept. 2006), T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervos, L. D. Whitley, and X. Yao, Eds., vol. 4193 of LNCS, Springer-Verlag, pp. 860–869.
- Cleary, R., and O’Neill, M. An attribute grammar decoder for the 01 multiconstrained knapsack problem. In Evolutionary Computation in Combinatorial Optimization – Evo- COP 2005 (Lausanne, Switzerland, 30 Mar.-1 Apr. 2005), G. R. Raidl and J. Gottlieb, Eds., vol. 3448 of LNCS, Springer Verlag, pp. 34–45.
- Cramer, N. L. A representation for the adaptive generation of simple sequential pro- grams. In Proceedings of the International Conference on Genetic Algorithms and Their Applications (Carnegie-Mellon University, Pittsburgh, PA, July 1985), J. J. Grefenstette, Ed., pp. 183–187.
- Da Costa, L. E., and Landry, J.-A. Relaxed genetic programming. In GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation (Seat- tle, Washington, USA, 8-12 July 2006), M. Keijzer et al., Ed., vol. 1, ACM Press, pp. 937– 938.
- Daida, J. M., Bertram, R., Stanhope, S., Khoo, J., Chaudhary, S., and Chaudhary, O. What makes a problem GP-hard? analysis of a tunably difficult problem in genetic programming. Genetic Programming and Evolvable Machines 2 (2001), 165–191.

References (contd.)

- Daida, J. M., Li, H., Tang, R., and Hilss, A. M. What makes a problem GP-hard? validating a hypothesis of structural causes. In Genetic and Evolutionary Computation – GECCO-2003 (2003), E. C.-P. et. al., Ed., vol. 2724 of LNCS, Springer-Verlag, Berlin, pp. 1665–1677.
- Darwin, C. On the Origins of the Species by Means of Natural Selection, or the Preser- vation of Favoured Races in the Struggle for Life. 1859.
- Deb, K., Horn, J., and Goldberg, D. Multimodal deceptive functions. Complex Systems 7 (1993), 131–153.
- Dempsey, I., O'Neill, M., and Brabazon, A. Constant creation with grammatical evolution. International Journal of Innovative Computing and Applications 1, 1 (2007), 23–38.
- Dempsey, I., O'Neill, M., and Brabazon, A. Foundations in Grammatical Evolution for Dynamic Environments, vol. 194 of Studies in Computational Intelligence. Springer, Apr. 2009.
- Eiben, A. E., and Jelasity, M. A critical note on experimental research methodology in EC. In Congress on Evolutionary Computation (CEC'02) (Honolulu, Hawaii, USA, 2002), IEEE Press, Piscataway, NJ, pp. 582–587.
- Eka´rt, A., and N´emeth, S. Z. Maintaining the diversity of genetic programs. In Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002 (Kinsale, Ire- land, 3-5 Apr. 2002), J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, Eds., vol. 2278 of LNCS, Springer-Verlag, pp. 162–171.
- Eklund, S. E. Time series forecasting using massively parallel genetic programming. In Proceedings of Parallel and Distributed Processing International Symposium (22-26 Apr. 2003), pp. 143–147.
- Evett, M., and Fernandez, T. Numeric mutation improves the discovery of numeric constants in genetic programming. In Genetic Programming 1998: Proceedings of the Third Annual Conference (University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998), J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, Eds., Morgan Kaufmann, pp. 66– 71.
- Fogel, D. Evolving computer programs. In Evolutionary Computation: The Fossil Record, D. Fogel, Ed. MIT Press, 1998, ch. 5, pp. 143–144.
- Fogel, L., Owens, A., and Walsh, M. Artificial Intelligence through Simulated Evolu- tion. John Wiley, 1966.
- Fonlupt, C. Solving the ocean color problem using a genetic programming approach. Applied Soft Computing 1, 1 (June 2001), 63–72.
- Francone, F. The discipulus owner's manual. URL: http://www.rmltech.com/technology overview.htm, 2004.

References (contd.)

- Francone, F. D., Nordin, P., and Banzhaf, W. Benchmarking the generalization capabilities of a compiling genetic programming system using sparse data sets. In Genetic Programming: Proceedings of the first annual conference (1996), J. R. Koza et al., Ed., MIT Press, Cambridge, pp. 72–80.
- Friedberg, R. A learning machine: Part 1. IBM J. Research and Development Vol. 2:1 (1958), 2–13.
- Friedberg, R., Dunham, B., and North, J. A learning machine: Part 2. IBM J. Research and Development (1959), 282–287.
- Gagne, C. Open beagle. URL: http://beagle.gel.ulaval.ca, 11 2007.
- Gagn´e, C., Schoenauer, M., Parizeau, M., and Tomassini, M. Genetic programming, validation sets, and parsimony pressure. In Genetic Programming, 9th European Conference, EuroGP2006 (2006), P. Collet et al., Ed., Lecture Notes in Computer Science, LNCS 3905, Springer, Berlin, Heidelberg, New York, pp. 109–120.
- Goldberg, D. E., and O'Reilly, U.-M. Where does the good stuff go, and why? how contextual semantics influence program structure in simple genetic programming. In Proceedings of the First European Workshop on Genetic Programming (Paris, 14-15 Apr. 1998), W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, Eds., vol. 1391 of LNCS, Springer-Verlag, pp. 16–36.
- Gustafson, S. An Analysis of Diversity in Genetic Programming. PhD thesis, School of Computer Science and Information Technology, University of Nottingham, Nottingham, England, Feb. 2004.
- Gustafson, S., and Vanneschi, L. Operator-based distance for genetic programming: Subtree crossover distance. In Genetic Programming, 8th European Conference, Eu- roGP2005 (2005), Keijzer, M., et al., Ed., Lecture Notes in Computer Science, LNCS 3447, Springer, Berlin, Heidelberg, New York, pp. 178–189.
- Gustafson, S., and Vanneschi, L. Operator-based tree distance in genetic program- ming. IEEE Transactions on Evolutionary Computation 12 (2008), 4.
- Hansen, J., Lowry, P., Meservy, R., and McDonald, D. Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. Decision Support Systems 43, 4 (2006), 1362–1374.
- Hemberg, E., Gilligan, C., O'Neill, M., and Brabazon, A. A grammatical genetic programming approach to modularity in genetic algorithms. In Proceedings of the 10th European Conference on Genetic Programming (Valencia, Spain, 11 - 13 Apr. 2007), M. Ebner, M. O'Neill, A. Eka´rt, L. Vanneschi, and A. I. Esparcia-Alca´zar, Eds., vol. 4445 of Lecture Notes in Computer Science, Springer, pp. 1–11.

References (contd.)

- Hornby, G. ALPS: the age-layered population structure for reducing the problem of premature convergence. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (2006), ACM New York, NY, USA, pp. 815–822.
- Hu, J., Goodman, E., Seo, K., Fan, Z., and Rosenberg, R. The hierarchical fair com- petition (hfc) framework for sustainable evolutionary algorithms. Evolutionary Compu- tation 13, 2 (2005), 241–277.
- Hu, T., and Banzhaf, W. Neutrality and variability: two sides of evolvability in linear genetic programming. In GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation (Montreal, 8-12 July 2009), G. Raidl, F. Rothlauf, G. Squillero, R. Drechsler, T. Stuetzle, M. Birattari, C. B. Congdon, M. Middendorf, C. Blum, C. Cotta, P. Bosman, J. Grahl, J. Knowles, D. Corne, H.-G. Beyer, K. Stanley, J. F. Miller, J. van Hemert, T. Lenaerts, M. Ebner, J. Bacardit, M. O'Neill, M. Di Penta, B. Doerr, T. Jansen, R. Poli, and E. Alba, Eds., ACM, pp. 963–970.
- Hu, T., and Banzhaf, W. The role of population size in rate of evolution in genetic programming. In Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009 (Tuebingen, Apr. 15-17 2009), L. Vanneschi, S. Gustafson, A. Moraglio, I. De Falco, and M. Ebner, Eds., vol. 5481 of LNCS, Springer, pp. 85–96.
- Jablonka, E., and Lamb, M. Evolution in four dimensions: Genetic, epigenetic, behav- ioral, and symbolic variation in the history of life. MIT Press, 2005.
- Jakobović, D., and Budin, L. Dynamic scheduling with genetic programming. In Pro- ceedings of the 9th European Conference on Genetic Programming (Budapest, Hungary, 10 - 12 Apr. 2006), P. Collet, M. Tomassini, M. Ebner, S. Gustafson, and A. Ekárt, Eds., vol. 3905 of Lecture Notes in Computer Science, Springer, pp. 73–84.
- Jonyer, I., and Himes, A. Improving modularity in genetic programming using graph- based data mining. In Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference (Melbourne Beach, Florida, USA, May 11-13 2006), G. C. J. Sutcliffe and R. G. Goebel, Eds., American Association for Artificial Intelligence, pp. 556–561.
- Kantschik, W., and Banzhaf, W. Linear-tree GP and its comparison with other GP structures. In Genetic Programming, Proceedings of EuroGP'2001 (Lake Como, Italy, 18-20 Apr. 2001), J. F. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, and W. B. Langdon, Eds., vol. 2038 of LNCS, Springer-Verlag, pp. 302–312.

References (contd.)

- Kantschik, W., and Banzhaf, W. Linear-graph GP—A new GP structure. In Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002 (Kinsale, Ire- land, 3-5 Apr. 2002), J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, Eds., vol. 2278 of LNCS, Springer-Verlag, pp. 83–92.
- Kashtan, N., and Alon, U. Spontaneous evolution of modularity and network motifs. Proceedings of the National Academy of Sciences 102, 39 (Sept. 27 2005), 13773–13778.
- Kashtan, N., Noor, E., and Alon, U. Varying environments can speed up evolution. Proceedings of the National Academy of Sciences 104, 34 (August 21 2007), 13711–13716.
- Katirai, H. Filtering junk E-mail: A performance comparison between genetic programming and naive bayes. 4A Year student project, 10 Sept. 1999.
- Keijzer, M., Babovic, V., Ryan, C., O'Neill, M., and Cattolico, M. Adaptive logic programming. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (San Francisco, California, USA, 7-11 July 2001), L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., Morgan Kaufmann, pp. 42–49.
- Keller, R. E., and Poli, R. Toward subheuristic search. In In Proceedings of 2008 IEEE Congress on Evolutionary Computation (2008), IEEE Press, pp. 3147–3154.
- Kinnear Jr., K. E. Fitness landscapes and difficulty in genetic programming. In Pro- ceedings of the First IEEEConference on Evolutionary Computing (1994), IEEE Press, Piscataway, NY, pp. 142–147.
- Kirschner, M., Gerhart, J., and Norton, J. The plausibility of life: Resolving Dar- win's dilemma. Yale Univ Pr, 2006.
- Kotanchek, M. The data modeler add-on package for mathematica. see http://www.evolved-analytics.com/ datamodeler, 72 2009.
- Koza, J. R. Hierarchical genetic algorithms operating on populations of computer programs. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89 (Detroit, MI, USA, 20-25 Aug. 1989), N. S. Sridharan, Ed., vol. 1, Morgan Kaufmann, pp. 768–774.
- Koza, J. R. A genetic approach to the truck backer upper problem and the inter-twined spiral problem. In Proceedings of IJCNN International Joint Conference on Neural Networks (1992), vol. IV, IEEE Press, pp. 310–318.
- Koza, J. R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA, 1992.

References (contd.)

- Koza, J. R. Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge Massachusetts, May 1994.
- Koza, J. R., Andre, D., Bennett III, F. H., and Keane, M. Genetic Programming 3: Darwinian Invention and Problem Solving. Morgan Kaufman, Apr. 1999.
- Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., and Lanza, G. Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, 2003.
- Kushchu, I. An evaluation of evolutionary generalization in genetic programming. Artificial Intelligence Review 18, 1 (2002), 3–14.
- Langdon, W. A many threaded cuda interpreter for genetic programming. In Proceed- ings of the 13th European Conference on Genetic Programming (2010), A. I. Esparcia-Alca´zar, A. Eka´rt, S. Silva, S. Dignum, and A. Uyar, Eds., vol. LNCS 6021, Springer, pp. 146–158.
- Langdon, W., and Banzhaf, W. Repeated patterns in genetic programming. Natural Computing 7, 4 (2008), 589–613.
- Langdon, W. B. Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!, vol. 1 of Genetic Programming. Kluwer, Boston, 24 Apr. 1998.
- Langdon, W. B., and Banzhaf, W. Genetic programming bloat without semantics. In Parallel Problem Solving from Nature - PPSN VI 6th International Conference (Paris, France, 16-20 Sept. 2000), M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., vol. 1917 of LNCS, Springer Verlag, pp. 201–210.
- Langdon, W. B., and Banzhaf, W. Repeated sequences in linear genetic programming genomes. Complex Systems 15, 4 (2005), 285–306.
- Langdon, W. B., and Banzhaf, W. Repeated patterns in genetic programming. Natural Computing 7, 4 (Dec. 2008), 589–613.
- Langdon, W. B., Gustafson, S., and Koza, J. R. GP Bibliography. http://www.cs.bham.ac.uk/ wbl/biblio/gp-bib-info.html, 2008.
- Langdon, W. B., and Poli, R. Genetic programming bloat with dynamic fitness. In Proceedings of the First European Workshop on Genetic Programming (Paris, 14-15 Apr. 1998), W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, Eds., vol. 1391 of LNCS, Springer-Verlag, pp. 96–112.
- Langdon, W. B., and Poli, R. Foundations of Genetic Programming. Springer-Verlag, 2002.

References (contd.)

- Lee, W.-C. Genetic programming decision tree for bankruptcy prediction. In Proceedings of the 2006 Joint Conference on Information Sciences, JCIS 2006 (Kaohsiung, Taiwan, ROC, Oct. 8-11 2006), Atlantis Press.
- Luke, S. ECJ. URL: http://cs.gmu.edu/ eclab/projects/ecj/, 2010.
- McConaghy, T., Leung, H., and Varadan, V. Functional reconstruction of dynamical systems from time series using genetic programming. In 26th Annual Conference of the IEEE Industrial Electronics Society, IECON 2000 (Nagoya, 22-28 Oct. 2000), vol. 3, IEEE, pp. 2031–2034.
- McKay, R. I. B., Hoai, N. X., Whigham, P. A., Shan, Y., and O'Neill, M. Grammar-based genetic programming a survey. Genetic Programming and Evolvable Machines (this issue) (2010).
- McPhee, N. F., Ohs, B., and Hutchison, T. Semantic building blocks in genetic programming. In Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008 (Naples, 26-28 Mar. 2008), M. O'Neill, L. Vanneschi, S. Gustafson, A. I. Esparcia Alcazar, I. De Falco, A. Della Cioppa, and E. Tarantino, Eds., vol. 4971 of Lecture Notes in Computer Science, Springer, pp. 134–145.
- Merelo, J., Keijzer, M., and Schoenauer, M. Eo evolutionary computation framework. URL: http:// eodev.sourceforge.net/, 2006.
- Mitchell, M., Forrest, S., and Holland, J. The royal road for genetic algorithms: fitness landscapes and ga performance. In Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life (1992), F. J. Varela and P. Bourgine, Eds., The MIT Press, pp. 245–254.
- Mitchell, T. Machine Learning. McGraw Hill, New York, 1996.
- Montana, D. J. Strongly typed genetic programming. Evolutionary Computation 3, 2 (1995), 199–230.
- Moore, J., Andrews, P., Barney, N., and White, B. Development and evaluation of an open-ended computational evolution system for the genetic analysis of susceptibility to common human diseases. Lecture Notes in Computer Science 4973 (2008), 129–140.
- Moore, J., Greene, C., Andrews, P., and White, B. Does Complexity Matter? Artifi- cial Evolution, Computational Evolution and the Genetic Analysis of Epistasis in common human Diseases. Genetic Programming Theory and Practice VI (2008), 125.
- Morrison, R. Designing Evolutionary Algorithms for Dynamic Environments. Springer, 2004.
- Nguyen, Q. U., Nguyen, T. H., Nguyen, X. H., and O'Neill, M. Improving the generalisation ability of genetic programming with semantic similarity based crossover. A. I. Esparcia-Alca´zar, A. Eka´rt, S. Silva, S. Dignum, and A. Uyar, Eds., vol. LNCS 6021, Springer, pp. 184–195.

References (contd.)

- Nguyen, Q. U., O'Neill, M., Nguyen, X. H., McKay, B., and Lopez, E. G. Semantic similarity based crossover in GP: The case for real-valued function regression. In Evolution Artificielle, 9th International Conference (26-28 Oct. 2009), P. Collet, Ed., Lecture Notes in Computer Science, pp. 13–24.
- Nicolau, M., Schoenauer, M., and Banzhaf, W. Evolving genes to balance a pole. A. I. Esparcia-Alca´zar, A. Ekart, S. Silva, S. Dignum, and A. Uyar, Eds., vol. LNCS 6021, Springer, pp. 196–207.
- Nordin, P., Banzhaf, W., and Francone, F. D. Introns in nature and in simulated structure evolution. In Bio-Computation and Emergent Computation (Skovde, Sweden, 1-2 Sept. 1997), D. Lundh, B. Olsson, and A. Narayanan, Eds., World Scientific Publishing.
- Oltean, M. Evolving evolutionary algorithms using linear genetic programming. Evo- lutionary Computation 13, 3 (Fall 2005), 387–410.
- O'Neill, M., and Brabazon, A. Recent patents in genetic programming. Recent Patents in Computer Science 2, 1 (2009), 43–49.
- O'Neill, M., McDermott, J., Swafford, J. M., Byrne, J., Hemberg, E., Shotton, E., McNally, C., Brabazon, A., and Hemberg, M. Evolutionary design using gram- matical evolution and shape grammars: Designing a shelter. International Journal of Design Engineering 3 (2010).
- O'Neill, M., and Ryan, C. Grammatical Evolution: Evolutionary Automatic Program- ming in a Arbitrary Language, vol. 4 of Genetic programming. Kluwer Academic Publishers, 2003.
- O'Reilly, U.-M., and Hemberg, M. Integrating generative growth and evolutionary computation for form exploration. Genetic Programming and Evolvable Machines 8, 2 (June 2007), 163–186. Special issue on developmental systems.
- Orfila, A., Estevez-Tapiador, J. M., and Ribagorda, A. Evolving high-speed, easy-to-understand network intrusion detection rules with genetic programming. In Applications of Evolutionary Computing, EvoWorkshops2009 (Tubingen, Germany, 15-17 Apr. 2009), M. Giacobini, I. De Falco, and M. Ebner, Eds., LNCS, Springer Verlag.
- P.Domingos. The role of Occam's razor in knowledge discovery. Data Mining and Knowledge Discovery 3, 4 (1999), 409–425.
- Poli, R., and Graff, M. There is a free lunch for hyper-heuristics, genetic programming and computer scientists. In Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009 (Tuebingen, Apr. 15-17 2009), L. Vanneschi, S. Gustafson, A. Moraglio, I. De Falco, and M. Ebner, Eds., vol. 5481 of LNCS, Springer, pp. 195–207.

References (contd.)

- Poli, R., Graff, M., and McPhee, N. F. Free lunches for function and program induction. In FOGA ’09: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms (Orlando, Florida, USA, 9-11 Jan. 2009), ACM, pp. 183–194.
- Poli, R., Langdon, W. B., and Holland, O. Extending particle swarm optimisation via genetic programming. In Proceedings of the 8th European Conference on Genetic Programming (Lausanne, Switzerland, 30 Mar. - 1 Apr. 2005), M. Keijzer, A. Tettamanzi, P. Collet, J. I. van Hemert, and M. Tomassini, Eds., vol. 3447 of Lecture Notes in Computer Science, Springer, pp. 291–300.
- Poli, R., Langdon, W. B., and McPhee, N. F. A field guide to ge- netic programming. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J. R. Koza).
- Poli, R., and McPhee, N. F. Exact schema theorems for GP with one-point and standard crossover operating on linear structures and their application to the study of the evolution of size. In Genetic Programming, Proceedings of EuroGP’2001 (2001), J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. Tettamanzi, and W. Langdon, Eds., vol. 2038 of LNCS, Springer-Verlag, pp. 126–142.
- Poli, R., and McPhee, N. F. General schema theory for genetic programming with subtree swapping crossover: Part I. Evolutionary Computation 11, 1 (2003), 53–66.
- Poli, R., and McPhee, N. F. General schema theory for genetic programming with
- subtree swapping crossover: Part II. Evolutionary Computation 11, 2 (2003), 169–206.
- Poli, R., and Vanneschi, L. Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms. In Genetic and Evolutionary Computation Conference,
- GECCO’07 (2007), D. Thierens et al., Ed., ACM Press, pp. 1335–1342.
- Poli, R., Vanneschi, L., Langdon, W. B., and McPhee, N. F. Theoretical results in genetic programming: The next ten years? Genetic Programming and Evolvable Machines (2010).
- Punch, B., Zongker, D., and Goodman, E. The royal tree problem, a benchmark for single and multiple population genetic programming. In Advances in Genetic Pro- gramming 2 (Cambridge, MA, 1996), P. Angeline and K. Kinnear, Eds., The MIT Press, pp. 299–316.
- Rissanen, J. Modeling by shortest data description. Automatica 14 (1978), 465–471.
- Rosca, J. P. Towards automatic discovery of building blocks in genetic programming. In Working Notes for the AAAI Symposium on Genetic Programming (1995), AAAI, pp. 78–85.

References (contd.)

- Rothlauf, F. Representations for genetic and evolutionary algorithms, second ed. Springer-Verlag, 2006. First published 2002, 2nd edition available electronically.
- Rothlauf, F., and Oetzel, M. On the locality of grammatical evolution. In Proceedings of the 9th European Conference on Genetic Programming (Budapest, Hungary, 10 - 12 Apr. 2006), P. Collet, M. Tomassini, M. Ebner, S. Gustafson, and A. Eka´rt, Eds., vol. 3905 of Lecture Notes in Computer Science, Springer, pp. 320–330.
- Ryan, C., and Keijzer, M. An analysis of diversity of constants of genetic programming. In Genetic Programming, Proceedings of EuroGP'2003 (Essex, 14-16 Apr. 2003), C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, Eds., vol. 2610 of LNCS, Springer-Verlag, pp. 404–413.
- Seront, G. External concepts reuse in genetic programming. In Working Notes for the AAAI Symposium on Genetic Programming (MIT, Cambridge, MA, USA, 10–12 Nov. 1995), E. V. Siegel and J. R. Koza, Eds., AAAI, pp. 94–98.
- Shekhar, S., and Amin, M. B. Generalization by neural networks. IEEE Trans. on Knowledge and Data Eng 4 (1992).
- Silva, S., and Vanneschi, L. Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction. In GECCO '09: Proceedings of the 11th An- nual conference on Genetic and evolutionary computation (Montreal, 8-12 July 2009), G. Raidl, F. Rothlauf, G. Squillero, R. Drechsler, T. Stuetzle, M. Birattari, C. B. Congdon, M. Middendorf, C. Blum, C. Cotta, P. Bosman, J. Grahl, J. Knowles, D. Corne, H.-G. Beyer, K. Stanley, J. F. Miller, J. van Hemert, T. Lenaerts, M. Ebner, J. Bac- ardit, M. O'Neill, M. Di Penta, B. Doerr, T. Jansen, R. Poli, and E. Alba, Eds., ACM, pp. 1115–1122.
- Silva, S. G. O. GPLab. A Genetic Programming Toolbox for MATLAB, 2008. See http://gplab.sourceforge.net.
- Smith, S. A learning system based on genetic adaptive algorithms.
- Smola A. J. and B. Scholkopf. A Tutorial on Support Vector Regression. Tech. Rep. Technical Report Series - NC2-TR-1998-030, NeuroCOLT2, 1999.
- Song, D., Heywood, M. I., and Zincir-Heywood, A. N. A linear genetic programming approach to intrusion detection. In Genetic and Evolutionary Computation – GECCO- 2003 (Chicago, 12-16 July 2003), E. Cantu´-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, Eds., vol. 2724 of LNCS, Springer-Verlag, pp. 2325–2336.

References (contd.)

- Spector, L. Evolving control structures with automatically defined macros. In Working Notes for the AAAI Symposium on Genetic Programming (MIT, Cambridge, MA, USA, 10–12 Nov. 1995), E. V. Siegel and J. R. Koza, Eds., AAAI, pp. 99–105.
- Spector, L., and Robinson, A. Genetic programming and autoconstructive evolution with the push programming language. Genetic Programming and Evolvable Machines 3, 1 (Mar. 2002), 7–40.
- Spencer, G. F. Automatic generation of programs for crawling and walking. In Advances in Genetic Programming, K. E. Kinnear, Jr., Ed. MIT Press, 1994, ch. 15, pp. 335–353.
- Stadler, P. F. Fitness landscapes. In Biological Evolution and Statistical Physics (Heidelberg, 2002), M. Lassig and Valleriani, Eds., vol. 585 of Lecture Notes Physics, Springer-Verlag, pp. 187–207.
- Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., and Tiwari, S. Problem definitions and evaluation criteria for the cec 2005 special session on real- parameter optimization. Tech. Rep. Technical Report Number 2005005, Nanyang Technological University, 2005.
- Teller, A., and Veloso, M. PADO: A new learning architecture for object recognition. In Symbolic Visual Learning, K. Ikeuchi and M. Veloso, Eds. Oxford University Press, 1996, pp. 81–116.
- Tomassini, M., Vanneschi, L., Collard, P., and Clergue, M. A study of fitness distance correlation as a difficulty measure in genetic programming. Evolutionary Com- putation 13, 2 (Summer 2005), 213–239.
- Vanneschi, L. Theory and Practice for Efficient Genetic Programming. PhD thesis, Faculty of Sciences, University of Lausanne, Switzerland, 2004.
- Vanneschi, L., Castelli, M., and Silva, S. Measuring bloat, overfitting and functional complexity in genetic programming. In GECCO ’10: Proceedings of the 12th Annual conference on Genetic and evolutionary computation (2010), J. Branke, Ed.
- Vanneschi, L., and Cuccu, G. Variable size population for dynamic optimization with genetic programming. In GECCO ’09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation (Montreal, 8-12 July 2009), G. Raidl, F. Rothlauf, G. Squillero, R. Drechsler, T. Stuetzle, M. Birattari, C. B. Congdon, M. Middendorf, C. Blum, C. Cotta, P. Bosman, J. Grahl, J. Knowles, D. Corne, H.-G. Beyer, K. Stanley, J. F. Miller, J. van Hemert, T. Lenaerts, M. Ebner, J. Bacardit, M. O’Neill, M. Di Penta, B. Doerr, T. Jansen, R. Poli, and E. Alba, Eds., ACM, pp. 1895–1896.
- Vanneschi, L., and Gustafson, S. Using crossover based similarity measure to improve genetic programming generalization ability. In GECCO ’09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation (New York, NY, USA, 2009), ACM, pp. 1139–1146.

References (contd.)

- Vanneschi, L., Gustafson, S., and Mauri, G. Using subtree crossover distance to in- vestigate genetic programming dynamics. In Genetic Programming, 9th European Con- ference, EuroGP2006 (2006), Collet, P., et al., Ed., Lecture Notes in Computer Science, LNCS 3905, Springer, Berlin, Heidelberg, New York, pp. 238–249.
- Vanneschi, L., Rochat, D., and Tomassini, M. Multi-optimization improves genetic programming generalization ability. In GECCO ’07: Proceedings of the 9th annual con- ference on Genetic and evolutionary computation (London, 7-11 July 2007), D. Thierens, H.-G. Beyer, J. Bongard, J. Branke, J. A. Clark, D. Cliff, C. B. Congdon, K. Deb, B. Do- err, T. Kovacs, S. Kumar, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, R. Poli, K. Sastry, K. O. Stanley, T. Stutzle, R. A. Watson, and I. Wegener, Eds., vol. 2, ACM Press, pp. 1759–1759.
- Vanneschi, L., Tomassini, M., Collard, P., and V´erel, S. Negative slope coefficient. A measure to characterize genetic programming. In Proceedings of the 9th European Conference on Genetic Programming (Budapest, Hungary, 10 - 12 Apr. 2006), P. Collet, M. Tomassini, M. Ebner, S. Gustafson, and A. Eka´rt, Eds., vol. 3905 of Lecture Notes in Computer Science, Springer, pp. 178–189.
- Vladislavleva, E. J., Smits, G. F., and den Hertog, D. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation 13, 2 (Apr. 2009), 333– 349.
- Wagner, A. Robustness and evolvability in living systems. Princeton University Press Princeton, NJ, 2005.
- Wagner, N., Michalewicz, Z., Khouja, M., and McGregor, R. Time series forecasting for dynamic environments: The dyfor genetic program model. IEEE Transactions on Evolutionary Computation 11, 4 (2006), 433–452.
- Wedge, D. C., and Kell, D. B. Rapid prediction of optimum population size in genetic programming using a novel genotype - fitness correlation. In GECCO ’08: Proceedings of the 10th annual conference on Genetic and evolutionary computation (Atlanta, GA, USA, 12-16 July 2008), M. Keijzer, G. Antoniol, C. B. Congdon, K. Deb, B. Doerr, N. Hansen, J. H. Holmes, G. S. Hornby, D. Howard, J. Kennedy, S. Kumar, F. G. Lobo, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, J. Pollack, K. Sastry, K. Stanley, A. Sto- ica, E.-G. Talbi, and I. Wegener, Eds., ACM, pp. 1315–1322.
- Weimer, W., Nguyen, T., Le Gues, C., and Forrest, S. Automatically finding patches using Genetic Programming. In International Conference on Software Engi- neering (ICSE) 2009 (2009), ACM New York, NY, USA, pp. 364–374.
- Whigham, P. A. Grammatical Bias for Evolutionary Learning. PhD thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 14 October 1996.

References (contd.)

- Whigham, P. A. Grammatically-based genetic programming. In Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications (Tahoe City, California, USA, 9 July 1995), J. P. Rosca, Ed., pp. 33–41.
- Wilson, G., and Heywood, M. Introducing probabilistic adaptive mapping developmental genetic programming with redundant mappings. Genetic Programming and Evolvable Machines 8, 2 (June 2007), 187–220. Special issue on developmental systems.
- Wolpert, D. H., and Macready, W. G. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1, 1 (1997), 67–82.
- Woodward, J. R. Modularity in genetic programming. In Genetic Programming, Proceedings of EuroGP'2003 (Essex, 14-16 Apr. 2003), C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, Eds., vol. 2610 of LNCS, Springer-Verlag, pp. 254– 263.
- Wright, S. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In Proceedings of the Sixth International Congress on Genetics (1932), D. Jones, Ed., vol. 1, pp. 355–366.
- Xie, H., Zhang, M., and Andreae, P. Genetic programming for automatic stress detection in spoken english. In Applications of Evolutionary Computing, EvoWorkshops2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoInteraction, EvoMUSART, EvoSTOC (Budapest, 10-12 Apr. 2006), F. Rothlauf, J. Branke, S. Cagnoni, E. Costa, C. Cotta, R. Drechsler, E. Lutton, P. Machado, J. H. Moore, J. Romero, G. D. Smith, G. Squillero, and H. Takagi, Eds., vol. 3907 of LNCS, Springer Verlag, pp. 460–471.
- Yang, S., Ong, Y.-S., and Jin, Y. Special issue on evolutionary computation in dynamic and uncertain environments. Genetic Programming and Evolvable Machines 7, 4 (2006).
- Zhang, M., Bhowan, U., and Ny, B. Genetic programming for object detection: A two-phase approach with an improved fitness function. Electronic Letters on Computer Vision and Image Analysis 6, 1 (2006), 27–43.