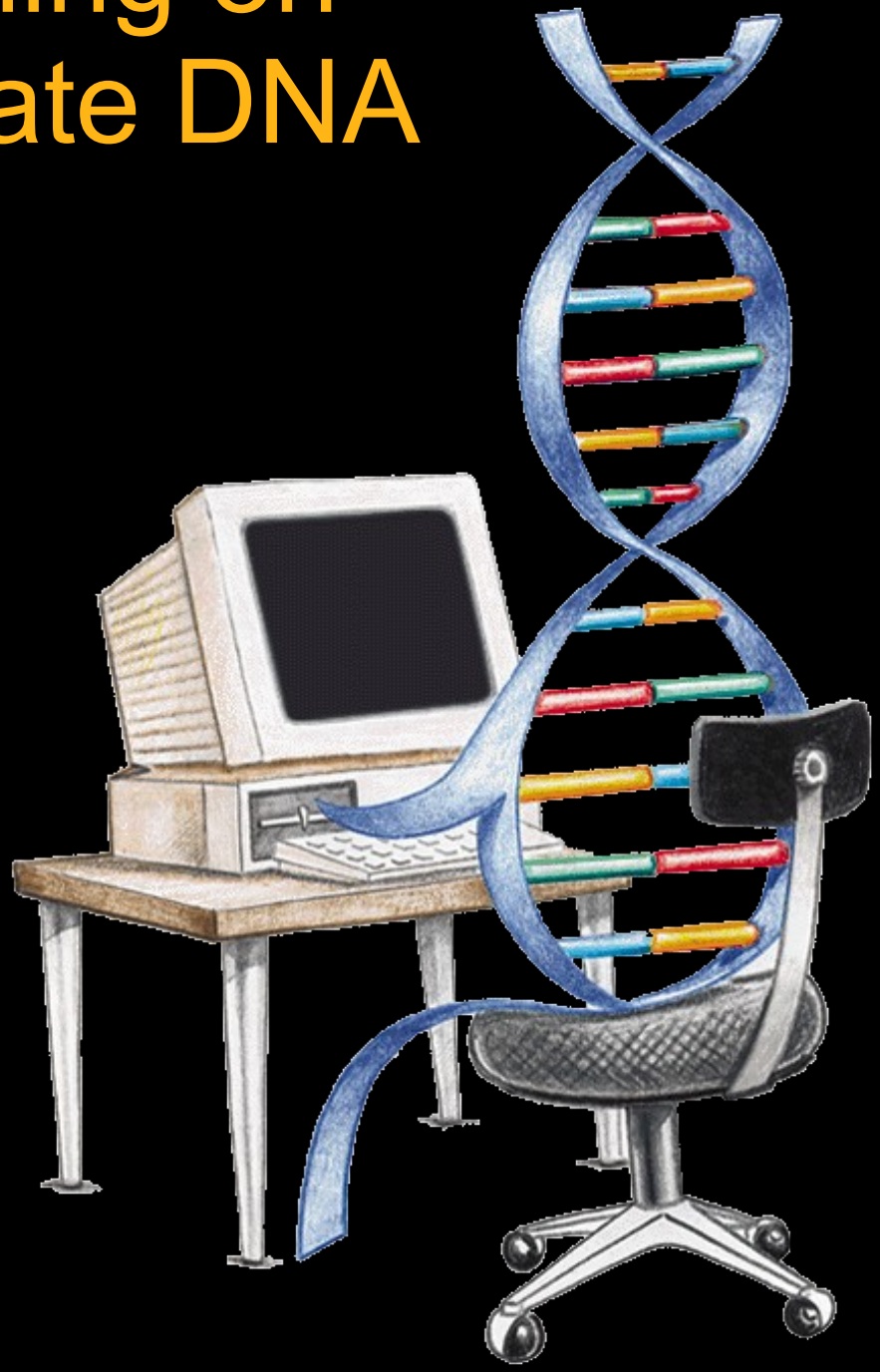
Genetic Programming on Grammars to generate DNA Motifs

Based on:
Automated DNA Motif Discovery
W. B. Langdon, O. Sanchez Graillet, A. P. Harrison
(arXiv:1002.0065v1)

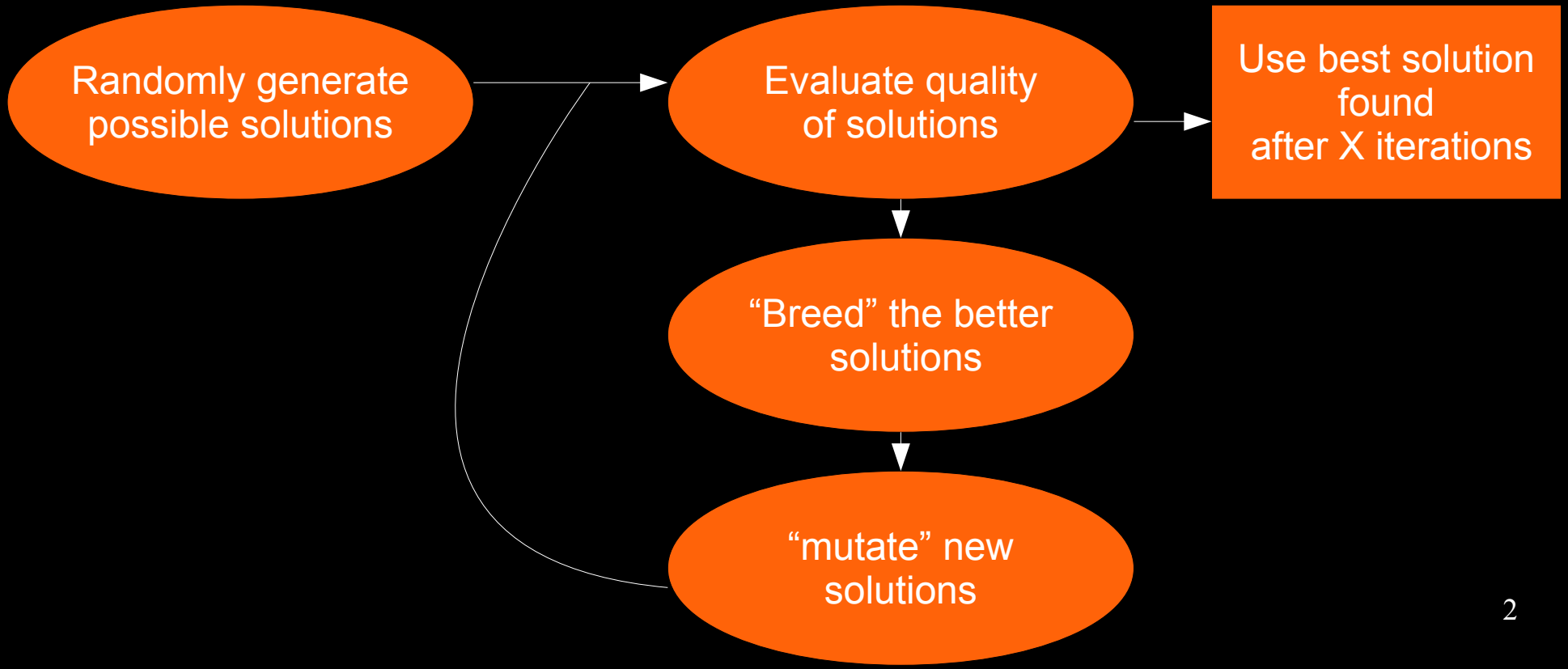


Thomas Unterthiner
0416441

Genetic Programming

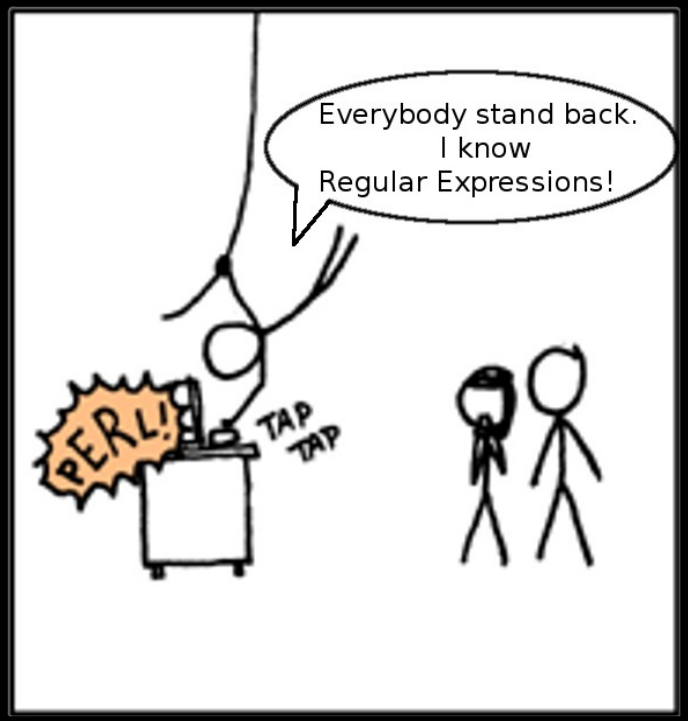
on Grammars to generate DNA Motifs

- Part of the “Evolutionary Algorithms”-family
  - EAs evolve potential “solutions” of a problem.
  - General working principle:



Genetic Programming on **Grammars** to generate DNA Motifs

- Grammars: formal languages to describe patterns in text
- Here: regular expressions (like in Perl)



[Image based on http://xkcd.com/208/]

Genetic Programming on Grammars to generate

DNA Motifs

- (biologically significant) Patterns in DNA
- Again: just think of Regular Expressions
- Here:
  - Distinguish between *non-coding* and *protein-coding* genes
  - Non-coding gene: pseudogene, snRNA, miRNA, ...

Materials & Methods

- 46.319 protein-coding and 9.836 non-coding transcripts (of 22.740 coding and 9.821 genes)
- Only take first 60 bases
- 50% training set, 50% test set

Running the GP

- The best 20% of the motifs of each iteration are used to generate the new iteration

- Performance-Measure:

  Σ (matched non-coding transcripts) +

  + Σ (not matched protein-transcripts)

- 50 iterations

Best Motif after 50 Iterations:

TACT|TGAT..|TA+TAT.|TA+(.CA+|T)(C|T)

- Performance on test set:

	Real non-protein	Real protein
Predicted non-protein	3683 (75%)	6884 (30%)
Predicted protein	1234 (25%)	16101 (70%)

Using only the first 60 bases

	Real non-protein	Real protein
Predicted non-protein	4529 (92%)	11207 (99%)
Predicted protein	382 (8%)	163 (1%)

Using the whole transcript

What I didn't like about the paper

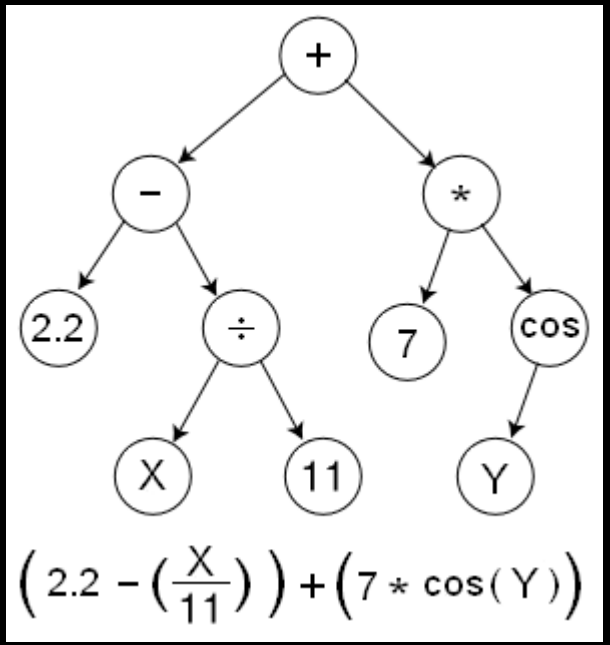
- Use of EA terminology without explaining it
- Only 1 run of the algorithm
  (a GP usually gives different results each time it's run)
- Give results on training-data
  (it's called training data for a reason!)
- Give results when using the whole transcript
  (not what the GP was trained for!)
- Did not compare results with other methods
  (They shortly mention that SVMs manage to get ~70% accuracy as well, but don't give any references!)

… that's all, folks

Any Questions?

Genetic Programming on Grammars to generate DNA Motifs (2)

- GP tries to evolve "programs" (or formulae)
- A possible solution can be represented as tree
- Trees can swap subtrees when "breeding"
- Nodes can change their value when mutating



Tree-representation of a possible solution to a problem like "find a function that best fits our measurement data"

Genetic Programming on **Grammars** to generate DNA Motifs (2)

- “Grammars” give you rules how to produce “Sentences”

- (stupid) Example:

```
<start> := <A> | <B>
<A> := “(“ <start> “)”
<B> = bbb | bbb<start> | <start>
```

  Can produce: “bbb(bbb)((bbb))”

- Can be represented as tree!

Grammar used to produce the REs

<start> ::= <RE>
<RE> ::= <union> | <simple-RE>
<union> ::= <RE> "|" <simple-RE>
<simple-RE> ::= <concatenation> | <basic-RE>
<concatenation> ::= <simple-RE> <basic-RE>
<basic-RE> ::= <RE-kleen> | <elementary-RE>
<RE-kleen>::= <minmaxquantifier> | <kleen>
<kleen>::= <star> | <plus>
<star> ::= <elementary-RE2> "*"
<plus> ::= <elementary-RE2> "+"
<minmaxquantifier> ::= <elementary-RE4> "{" <int> <optREint> "}"
<elementary-RE> ::= <group> | <elementary-RE1>
<elementary-RE1> ::= <xos> | <elementary-RE2>
<elementary-RE2> ::= <any> | <elementary-RE3>
<elementary-RE3>::= <set> | <char>
<elementary-RE4> ::= <group> | <elementary-RE2>
<group> ::= "(" <RE> ")"
<xos> ::= <sos> | "$"
<sos> ::= "^" <elementary-RE4>
<set> ::= <positive-set> | <negative-set>
<positive-set> ::= "[" <set-items> "]"
<negative-set> ::= "[^" <set-items> "]"

<set-items> ::= <set-item> | <set-items2>
<set-items2> ::= <set-item> <set-items>
<set-item> ::= <char>
<char> ::= <c00> | <c01>
<any> ::= "."
<c00> ::= T | C
<c01> ::= A | G
<optREint> ::= <2ndint> | $
<2ndint> ::= "," <int>
<int> ::= <d0>

#4 Bit Gray Code Encoder
<REdigit> ::= <d111> | <d0>
<d0> ::= <d00> | <d01>
<d00> ::= <d000> | <d001>
<d01> ::= <d010> | <d011>
<d000> ::= 1
<d001> ::= 3 | 2
<d010> ::= 7 | 6
<d011> ::= 4 | 5
<d111> ::= 8 | 9

Example of a sentence