Stochastic optimization algorithms
Lecture 8, 20200916

Evolutionary algorithms:
LGP and IEC

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

Today's learning goals

- After this lecture you should be able to
  - Describe linear genetic programming (LGP) and discuss the differences between LGP and GAs.
  - Apply LGP in a function fitting task.
  - Describe interactive evolutionary computation (IEC)

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Genetic programming

- Evolves computer programs (varying size)
- (Standard) genetic programming (GP): Applied to tree-like structures:

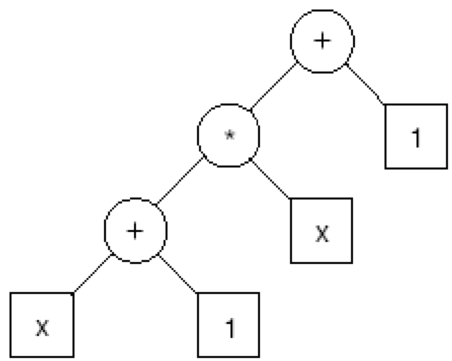


Figure 3.16: *A GP tree, which can be evaluated to give* $f(x) = (x+1)x+1$.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Genetic programming

- Evolves computer programs (varying size)
- (Standard) genetic programming (GP): Applied to tree-like structures:

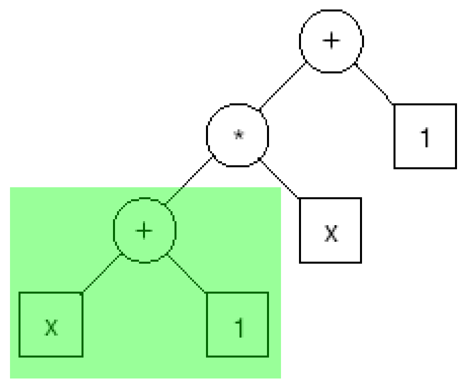


Figure 3.16: *A GP tree, which can be evaluated to give* $f(x) = (x+1)x + 1$.

Genetic programming

- Evolves computer programs (varying size)
- (Standard) genetic programming (GP): Applied to tree-like structures:

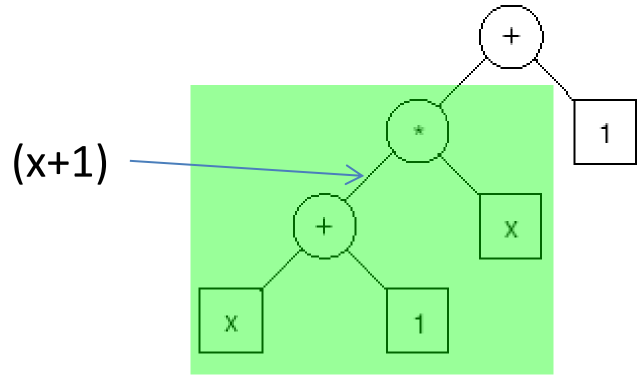


Figure 3.16: *A GP tree, which can be evaluated to give* $f(x) = (x+1)x+1$.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Genetic programming

- Evolves computer programs (varying size)
- (Standard) genetic programming (GP): Applied to tree-like structures:

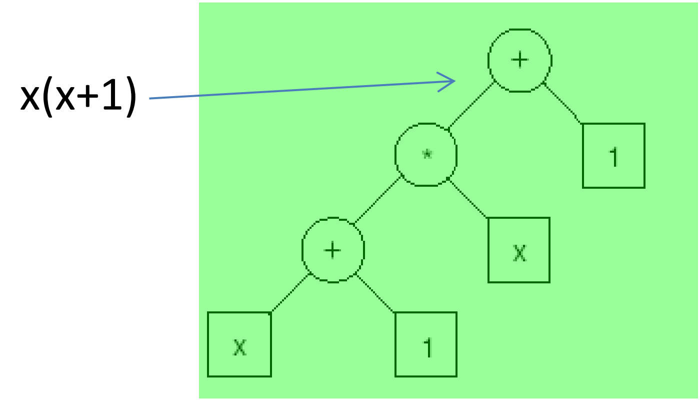
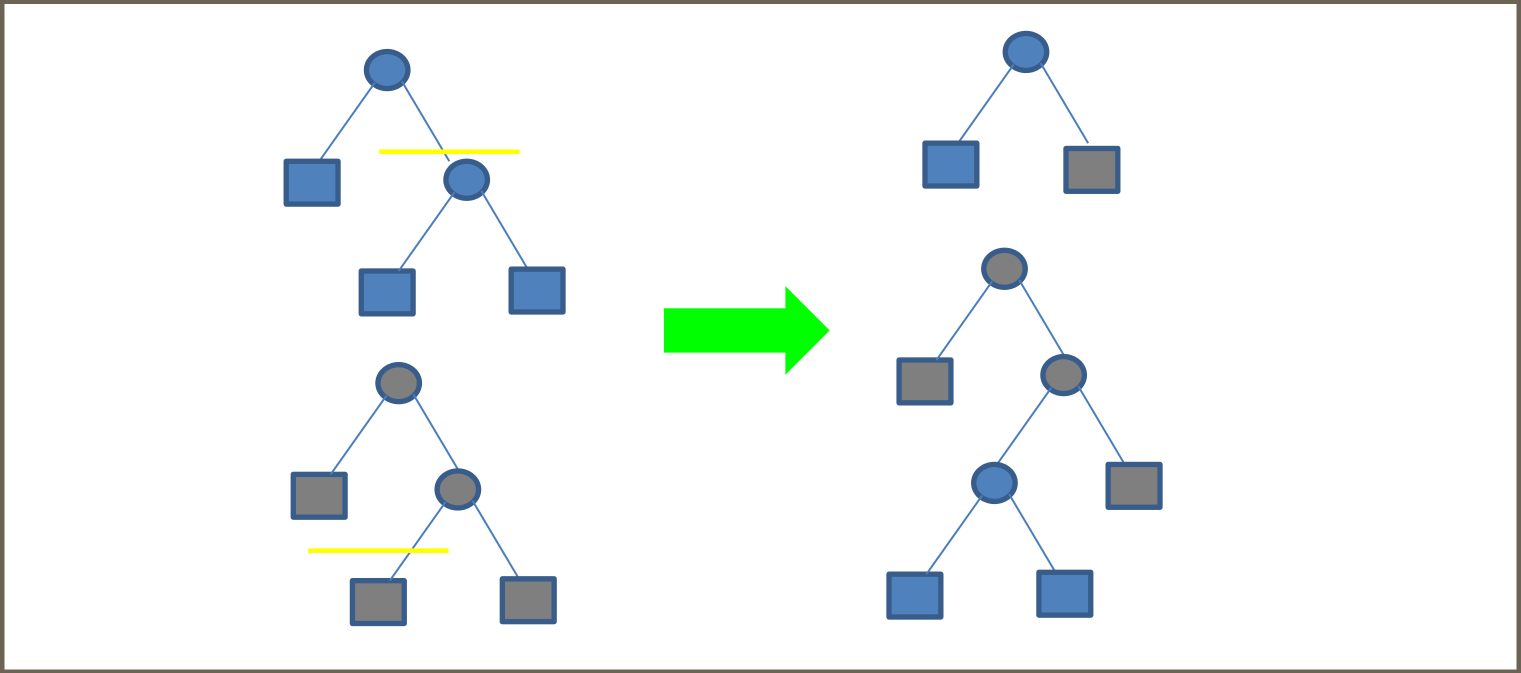


Figure 3.16: *A GP tree, which can be evaluated to give* $f(x) = (x+1)x+1$.

Genetic programming

- Crossover



Linear genetic programming

- **Linear genetic programming** (LGP): Evolves computer programs consisting of sequences of elementary instructions. Here we shall only consider LGP.

Registers and instructions

- Registers and instructions are <u>central concepts</u> in LGP.
- Registers: Two kinds:
  - **Variable registers** r_i (storing variables)
  - **Constant registers** c_i (storing constants, set once and for all)
- Instructions:
  - Set of allowed operators = **instruction set** (e.g {+,-,*,/})

Registers and instructions

- In LGP one evolves sequences of operations of the form

<Destination register> := <operand1> <operator> <operand2>

- Example: $r_1 := r_2 + c_1$
- At initialization, set constant registers once and for all.
- New constants can be built (and stored in variable registers) during a run.

Registers and instructions

- The instruction set can consist of basic mathematical operators such as +,-,*,/, ...
- ...but also (for example) sin(), exp() etc. in which case only the first operand is used (the second operand is ignored).
- **Protected definitions** (of instructions): Makes sure that all chromosomes can be evaluated. See Eq. (3.52), p. 74.
- Other types of instructions are also possible. For example, in a robot task, one may have operators such as TurnLeft(), TurnRight(), MoveForward(), Stop(), Grasp(), Lift() etc.

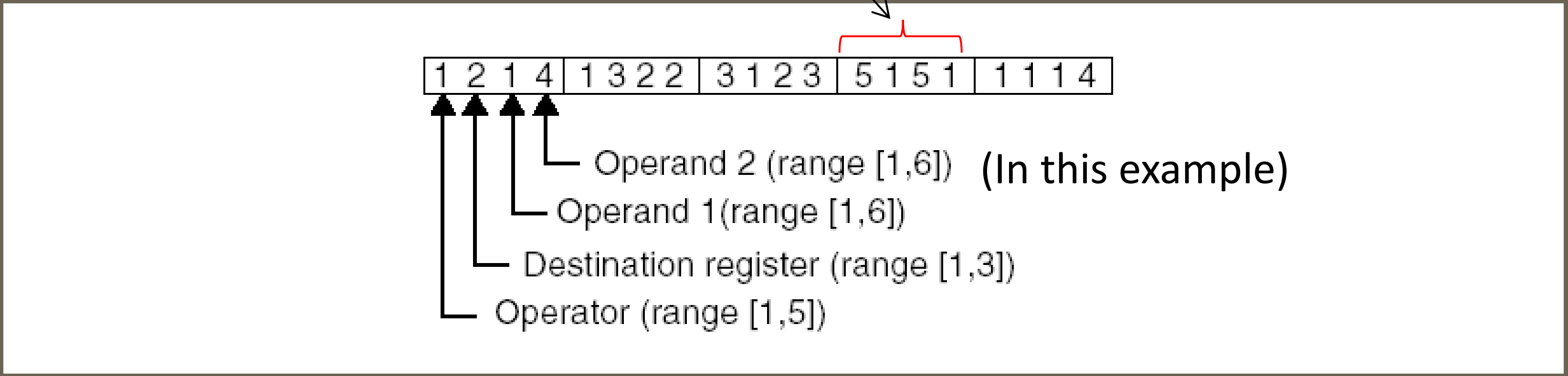
Decoding and evaluation

- Consider the example in Fig. 3.19 in the book, where there are
  - 3 variable registers $\mathcal{R} = (r_1, r_2, r_3)$
  - 3 constant registers $\mathcal{C} = (c_1, c_2, c_3)$ = (1,3,10)
- The destination registers are chosen from the set $\mathcal{R}$, whereas the operands are chosen from the set $\mathcal{A} = \mathcal{R} \cup \mathcal{C} = (r_1, r_2, r_3, c_1, c_2, c_3)$
- The operators are chosen from the set $\mathcal{O} = (o_1, o_2, o_3, o_4, o_5)$=(+,-,*,/, >), where ">" is the operator "is larger than"

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Decoding and evaluation

MISPRINT! (Also in Fig. 3.20): Should be: 5 1 1 5



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Decoding and evaluation

- Consider a case where the chromosome represents some function $f(x)$.
- To evaluate the function, set $r_1 = x$, and $r_2 = r_3 = 0$, then run through the chromosome (i.e. compute the sequence of operations).
- At the end, take the content of some predetermined register (e.g. r_1) as the estimate of $f(x)$ (denoted $\hat{f}(x)$).
- To fit a function, one must therefore run through the LGP chromosome many times, once for each value of x.

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Initially:
$r_1 = x = 1,$
$r_2 = 0, r_3 = 0$

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Operator index (from the set $\mathcal{O}$)

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Destination register index (from the set $\mathcal{R}$)

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Operand indices (from the set $\mathcal{A}$)

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

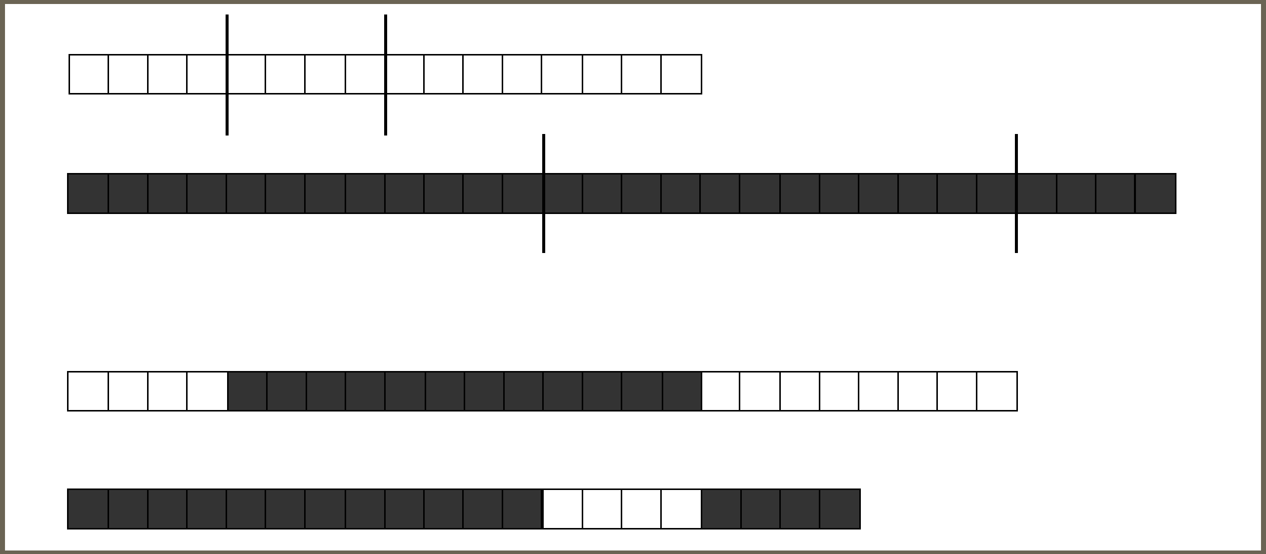
Decoding and evaluation

- Specific example (computed for $x = 1$), see also the chromosome shown earlier

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 1, 5	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Evolutionary operators

- In LGP, selection takes place just as in a GA.
- Crossover in LGP, however, uses two crossover points and is non-length preserving:



Evolutionary operators

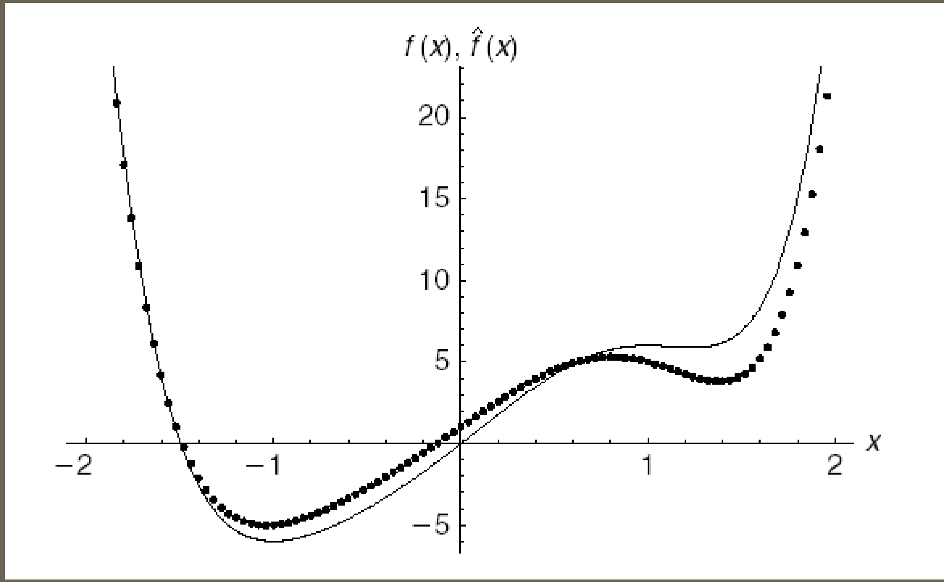
- Mutations are essentially as in a GA, but one must take into account that different genes may have different range, depending on whether they encode a destination register, an operand, or an operator.

Today’s learning goals

- After this lecture you should be able to
  - Describe linear genetic programming (LGP) and discuss the differences between LGP and GAs.
  - Apply LGP in a function fitting task.
  - Describe interactive evolutionary computation (IEC)

Function fitting with LGP

- Consider Example 3.11.
- The aim is to find (using LGP) the functional form of a function $f(x)$, based on a set of samples of that function.



Function fitting with LGP

- Procedure:
  - Choose instruction set (e.g. {+,-,*,/})
  - Set constant registers (once and for all)
  - Initialize chromosomes.
  - For each chromosome, run through all data points $x_i, i = 1, \ldots L$.
    - For each point, place the value of x_i in (for example) the first register, and set the other registers to 0.
    - Run the LGP chromosome, and the take the contents (after evaluation) of (for example) the first register as the estimate $\hat{f}(x_i)$ of the function value $f(x_i)$… (continued on next page)

Function fitting with LGP

- Once all points have been considered, form the error as (for example) …

$$E_{\mathrm{rms}} = \sqrt{\frac{1}{L}\sum_{i=1}^{L}\left(f(x_i) - \hat{f}(x_i)\right)^2}$$

- …and then compute the fitness as $F = 1/E_{\mathrm{rms}}$.

- Then, apply the evolutionary operators as usual etc.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Symbolical evaluation

- Instead of inserting a numerical value, insert the symbol x in $r_1 = x$, and set $r_2 = r_2$= 0.
- Then evaluate as usual. For example, the first two instruction in Figs. 3.19 and 3.20 would give:

$$r_2 := r_1 + c_1 \Rightarrow r_2 = \mathrm{x} + 1$$
$$r_3 := r_2 + r_2 \Rightarrow r_3 = 2\mathrm{x} + 2$$

- ..and so on.
- In Matlab, one can use the syms toolbox or use string operations, representing registers and operators as string (e.g. as in ‘x’ ‘+’ ‘1’ = ‘x+1’ etc.)

Today’s learning goals

- After this lecture you should be able to
  - Describe linear genetic programming (LGP) and discuss the differences between LGP and GAs.
  - Apply LGP in a function fitting task.
  - Describe interactive evolutionary computation (IEC)

Interactive evolutionary computation

- In some cases (e.g. design problems) one can use *subjective* evaluation of individuals (rather than (just) an objective fitness function).
- Used in **interactive evolutionary computation** (IEC).

Interactive evolutionary computation

- Basic algorithm: (see also Fig. 3.23 in the book)

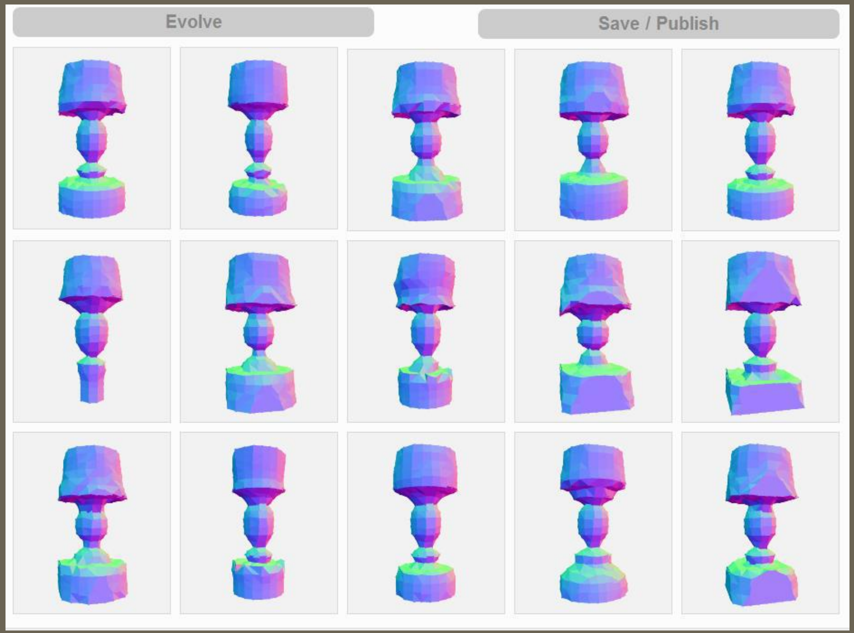
1. Initialize the population by generating (from randomly initialized chromosomes) and displaying N images on a computer screen.
2. Subjectively evaluate the individuals and select the preferred image.
3. Generate the next generation:
   - 3.1. Make an exact copy of the preferred image and place it at the centre of the screen.
   - 3.2. Make $N - 1$ new individuals by mutating the selected individual. Display the $N - 1$ newly generated individuals, surrounding the individual placed at the centre of the screen.
4. Return to step 2, unless any of the displayed images are satisfactory.

Interactive evolutionary computation

- More advanced versions also involve crossover etc.
- Constraint handling (where applicable) can also be implemented, so that the user is only presented with solution candidates that fulfil the constraints, if any.
- Main difficulties:
  - Finding a suitable encoding for the problem at hand.
  - User fatigue.

Interactive evolutionary computation

- Example, See http://endlessforms.com/



Today's learning goals

- After this lecture you should be able to
  - Describe linear genetic programming (LGP) and discuss the differences between LGP and GAs.
  - Apply LGP in a function fitting task.
  - Describe interactive evolutionary computation (IEC)