Gene Expression Programming

Jiří Kubalík Department of Cybernetics, CTU Prague

Substantial part of this material is based on the article Candida Ferreira: Gene Expression Programming: A New Adaptive Algorithm for Solving, see http://arxiv.org/ftp/cs/papers/0102/0102027.pdf



http://cw.felk.cvut.cz/doku.php/courses/a0m33eoa/start

Contents

Gene Expression Programming

- Representation
- Variation operators
- Automatically Defined Functions
- Examples: Symbolic regression

Gene Expression Programming (GEP) - genotype/phenotype genetic algorithm for creation of computer programs

- GEP uses **fixed length linear chromosomes** of specific structural organization of genes.
- The chromosomes are subjected to variation operators.
- The linear chromosomes are expressed as nonlinear expression trees (ETs) of different sizes and shapes that are evaluated and upon which the selection acts.

Any modification made in the genome always results in syntactically correct ETs (*given that the closure property holds*).

• GEP provides means for automatic defining and reusing functions.

GEP recalls to its analogy to the natural **gene expression**:

Gene expression is the process by which information from a gene is used in the synthesis of a functional gene product. These products are often proteins, but in non-protein coding genes such as rRNA genes or tRNA genes, the product is a functional RNA.

Wikipedia

A chromosome consists of a linear string of fixed length composed of one or more genes.

Within each gene, a coding sequences of symbols, terminals and functions, (open reading frame - ORF) can be followed by noncoding region.

ORFs are K-expressions that are translated into ETs using a breadth-first parsing:

input: K - expression (from Karva language) output: ET – expression tree o – current open node, s – currently processed symbol of K Read the first symbol s of K 1. 2. Make an empty root node of the ET 3 $o \leftarrow root$ 4. Attach the corresponding function/terminal, S, to the node o 5 and create bellow the node as many empty children nodes as there are arguments to that function, S. 6. If the ET is complete then end. 8. Else 9. Read next symbol s from K $o \leftarrow left_upper_open_node(ET)$ 10. 11. Goto line 4.

Assume K-expression

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 Q * + * a * Q a a b a b b a a b

where Q denotes the square root function, and 'a' and 'b' are variables.

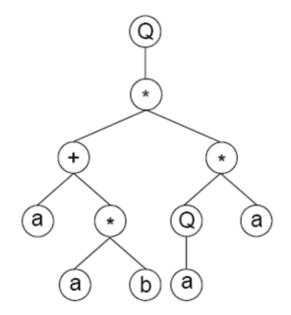
Then the following $\ensuremath{\mathsf{ET}}$ is

Assume K-expression

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
Q	*	+	*	а	*	Q	a	a	b	a	b	b	а	а	b

where Q denotes the square root function, and 'a' and 'b' are variables.

Then the following $\ensuremath{\mathsf{ET}}$ is



Organization of genes

• **Head** – contains both the function and terminal symbols.

The head length, h, is chosen by the user for each problem.

Tail – contains only terminal symbols.

The length of the tail t is a function of h and the number of arguments of the function with the biggest arity.

The tail must be long enough to ensure that there are sufficient number of terminals for all arguments induced by functions present in the head part.

Consider $T = \{a, b\}$, $F = \{Q, +, -, *, /\}$ and a gene (the tail is shown in red):

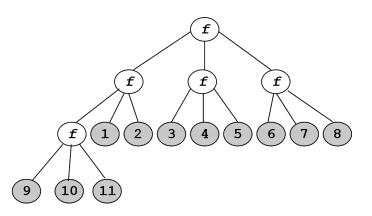
0 1 2 3 4 5 6 7 8 9 0 | 1 2 3 4 5 6 7 8 9 0 + Q - / b * a a Q b a a b a a b b a a a b

Consider

- Head length h = 5,
- The maximum arity over all functions, n = 3.

and assume that all of the symbols in the head part represent functions with arity n = 3.

What is the number of open nodes that must be filled in by terminal symbols?

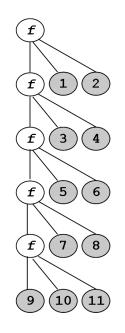


Consider

- Head length h = 5,
- The maximum arity of functions n = 3.

and assume that all of the symbols in the head part represent functions with arity n = 3.

What is the number of open nodes that must be filled in by terminal symbols?

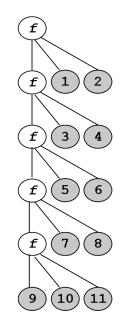


Consider

- Head length h = 5,
- The maximum arity of functions n = 3.

and assume that all of the symbols in the head part represent functions with arity n = 3.

What is the number of open nodes that must be filled in by terminal symbols?



The size of the tail t must be

$$t = h(n-1) + 1$$

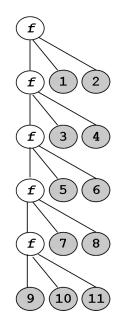
in order to ensure a sufficient number of terminal symbols even for the worst case scenario with only function symbols in the head part.

Consider

- Head length h = 5,
- The maximum arity of functions n = 3.

and assume that all of the symbols in the head part represent functions with arity n = 3.

What is the number of open nodes that must be filled in by terminal symbols?



The size of the tail t must be

$$t = h(n-1) + 1$$

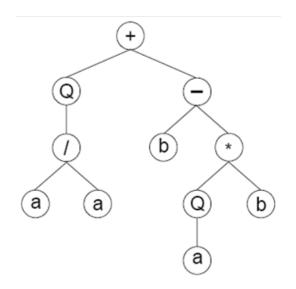
in order to ensure a sufficient number of terminal symbols even for the worst case scenario with only function symbols in the head part.

Note, the length of the ORFs varies, not the length of genes.

 the noncoding regions in genes allow modifications of the genome always producing syntactically correct programs. Consider $T = \{a, b\}$, $F = \{Q, +, -, *, /\}$ and a gene (the tail is shown in red):

0 1 2 3 4 5 6 7 8 9 0 | 1 2 3 4 5 6 7 8 9 0 + Q - / b * a a Q b a | a b a a b b a a a b

that codes for the following ET:



In this case the ORF ends at position 9.

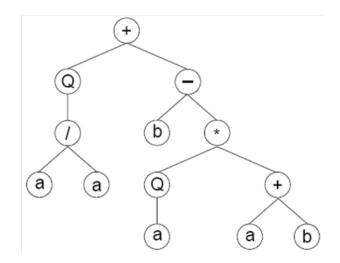
GEP: Effect of Variable ORF Size

Suppose now the symbol at position 9 changed from 'b' into '+'.

The following gene will be:

0 1 2 3 4 5 6 7 8 9 0 1 2 | 3 4 5 6 7 8 9 0 + Q - / b * a a Q + a a b | a a b b a a a b

that codes for the following ET:

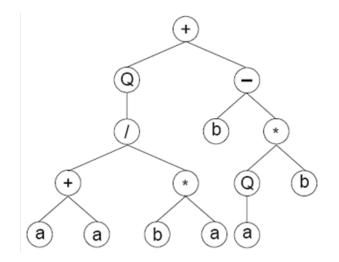


In this case the ORF ends at position 12.

Suppose now the symbols at positions 6 and 7 (in the original gene) changed into '+' and '*'. The following gene will be:

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 | 5 6 7 8 9 0 + Q - / b * + * Q b a a b a a | b b a a a b

that codes for the following ET:



In this case the ORF ends at position 14.

Despite its fixed length, each gene can code for ETs of different sizes and shapes!

Chromosomes are usually composed of more than one gene of equal length – the number of genes is the control parameter defined by a user.

Each gene codes for a sub-ET

- 1. The sub-ETs can interact with one another and form a more complex ET.
 - (a) The sub-ETs are linked together by a linking function. The linking function can be either defined for a problem at hand or evolved along with the genes.
 - (b) The sub-ETs are linked together by means of so-called **homeotic genes** genes representing the "main program".

The homeotic genes are evolved to determine which sub-ETs are called upon and how the sub-ETs interact with one another (**the concept similar to ADFs in GP**).

2. Each sub-ET can define one individual output in multi-output problems (for example, each sub-ET can be responsible for an identification to a particular class in classification problems).

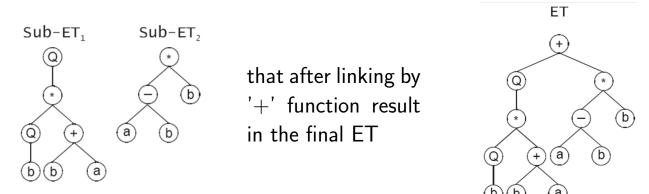
Multigenic Chromosomes: Linking Functions

Using an addition linking function '+' for algebraic sub-ETs.

A two-genic chromosome:

0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 Q * Q + b b a a a * - b a b a a b b

expresses as two sub-ETs

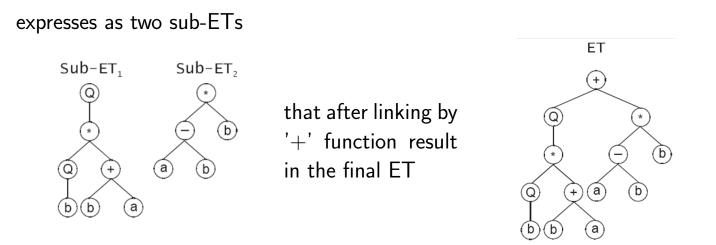


Multigenic Chromosomes: Linking Functions

Using an addition linking function '+' for algebraic sub-ETs.

A two-genic chromosome:

0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 Q * Q + b b a a a * - b a b a a b b



Note that the final ET could be encoded by a single-genic chromosome

0 1 2 3 4 5 6 7 8 9 0 1 2 + Q * * - b Q + a b b b a

however, multigenic chromosomes, allowing for modular construction of complex, hierarchical structures are more efficient.

A chromosome is composed of

- one or more conventional genes that act as ADFs,
- plus so called homeotic gene the expression of such genes results in the main program that determines which genes are expressed and how the corresponding sub-ETs (ADFs) interact with each another.

Homeotic genes have exactly the same structural organization as conventional genes – they have specific length and specific sets of functions and terminals.

- Head contains linking functions and so-called genic terminals representing conventional genes.
- **Tail** contains only **genic terminals**.

The concept using homeotic genes allows for encoding the ADFs that can be further called many times from many different places in the "main program".

Contrary to GP, the ADFs in GEP do not allow formal parameters to be defined.

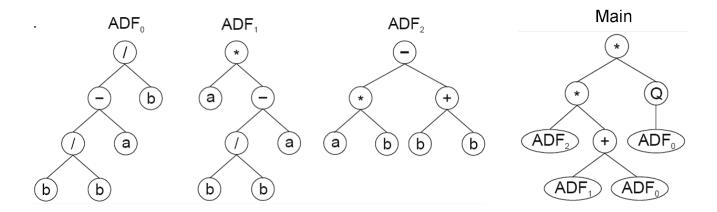
Example:

- the head length of the homeotic gene $h_H = 5$
- $\hfill \hfill \hfill$
- $F_H = \{+, -.*, /, Q\}$, $T_H = \{1, 2, 3\}$ denoting ADF_1 , ADF_2 and ADF_3
- $F = \{+, -, *, /\}$ and $T = \{a, b\}$

The following chromosome codes for three conventional genes and one homeotic gene (red)

01234567801234567801234567801234567890 /-b/abbaa*a-/abbab-*+abbbaa**Q2+010102

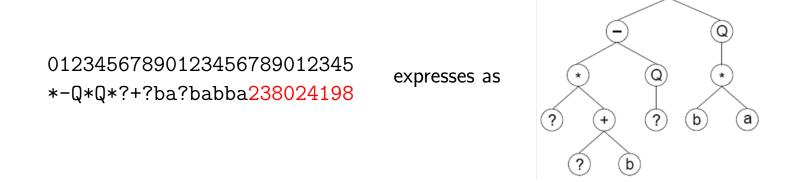
that expresses as



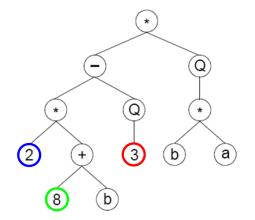
Handling random numerical constants in GEP:

- Extra terminal '?' represents the random constant (similar to the *ephemeral random constant* in GP).
- Extra domain Dc comes after the tail, has a length equal to t and is composed of symbols chosen to represent the random constants.
- For each gene the initial random constants are generated during the creation of the initial population and kept in an array. Specific operators to introduce genetic variation in the pool of random constants and for effective circulation of the random constants in the population.

Example: The following single-gene chromosome with h = 8 (the Dc is shown in red)



Then the ?'s in the ET are replaced from left to right and from top to bottom by the symbols in Dc



The random constants corresponding to these symbols are kept in an array; the number represented by the numeral indicates the order in the array

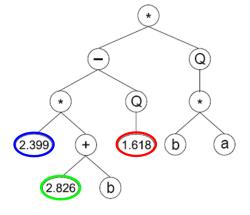
 $A = \{1.095, 1.816, 2.399, 1.618, 0.725, 1.997, 0.094, 2.998, 2.826, 2.057\}.$

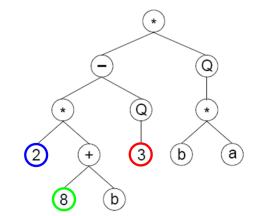
Then the ?'s in the ET are replaced from left to right and from top to bottom by the symbols in Dc

The random constants corresponding to these symbols are kept in an array; the number represented by the numeral indicates the order in the array

 $A = \{1.095, 1.816, 2.399, 1.618, 0.725, 1.997, 0.094, 2.998, 2.826, 2.057\}.$

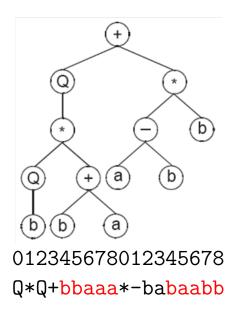
Finally, after substituting numerals for the actual values we get



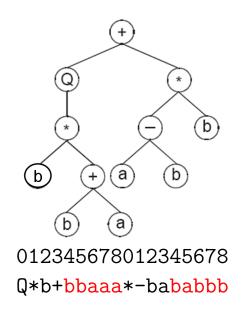


Mutation – changes a symbol anywhere in the chromosome.

- The desired structural organization of chromosomes must be ensured.
 In the heads, any symbol can change to another (function or terminal).
 In the tails, only terminals can be introduced.
- Neutral mutations are possible when it occurs in the noncoding region.



A mutation occurred at position 2 in gene 1 and at position 6 in gene 2.



Transposition – certain fragments of the genome can be copied/moved to another place in the chromosome.

Three types of transposable elements

 Insertion sequence elements (IS) – short fragments with a function or terminal in the first position that transpose to the head of genes (except to the root).

This transposition can drastically reshape the ET (the closer to the gene beginning the insertion site is the more profound the change).

 Root IS elements – short fragments with a function in the first position that transpose to the root of genes.

Modifications introduced by this transposition are extremely radical – useful for creating genetic variation.

• Entire genes – an entire gene transposes itself to the beginning of the chromosome and is deleted in the place of origin.

Useful only in situations where the linking function is not commutative.

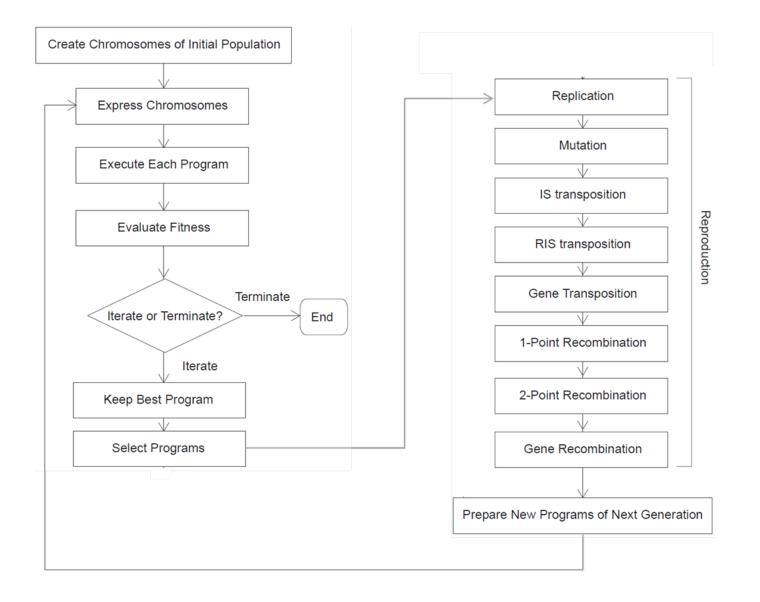
In all types of transposition the structural organization of chromosomes is maintained.

GEP Genetic Operators: Recombination

Three type of recombination:

- 1-point recombination, 2-point recombination
- Gene recombination

GEP: Flowchart



GEP on Symbolic Regression

Target function:
$$y = a^4 + a^3 + a^2 + a$$

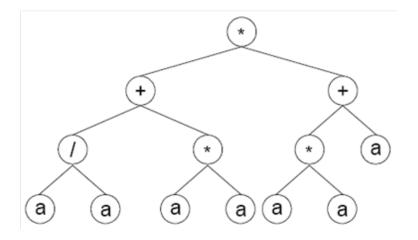
GEP

- $F = \{+, -, *, /\}$
- $T = \{a\}$

Example solution found with h = 6

A single-genic chromosome

0123456789012 expresses as *++/**aaaaaaa



Application areas

- Symbolic regression
- Classification problems
- Logic synthesis
- Design of neural networks

Pros/cons:

- (+) Structural organization of genotypes provides an efficient way of encoding syntactically correct programs.
- (+) Efficient means for preventing bloat.
- (-) ADFs can only be reused with the same inputs.
- (-) It is not possible to evolve programs with function nodes of different output type.
- (-) Requires proper setting of control parameters -h and the number of genes.

- Gene Expression Programming website (Candida Ferreira, Gepsoft Limited): http://www.gepsoft.com/
- Candida Ferreira: Gene Expression Programming: A New Adaptive Algorithm for Solving, Complex Systems, Vol. 13, issue 2: 87-129, 2001 http://arxiv.org/ftp/cs/papers/0102/0102027.pdf