Deriving Trading Rules Using Gene Expression Programming

Adrian VISOIU
Academy of Economic Studies Bucharest - Romania
Economic Informatics Department - collaborator
adrian.visoiu@csie.ase.ro

This paper presents how buy and sell trading rules are generated using gene expression programming with special setup. Market concepts are presented and market analysis is discussed with emphasis on technical analysis and quantitative methods. The use of genetic algorithms in deriving trading rules is presented. Gene expression programming is applied in a form where multiple types of operators and operands are used. This gives birth to multiple gene contexts and references between genes in order to keep the linear structure of the gene expression programming chromosome. The setup of multiple gene contexts is presented. The case study shows how to use the proposed gene setup to derive trading rules encoded by Boolean expressions, using a dataset with the reference exchange rates between the Euro and the Romanian leu. The conclusions highlight the positive results obtained in deriving useful trading rules.

Keywords*: Gene Expression Programming, Trading Rules, Genetic Algorithms, Exchange Rate*

1 Introduction

Trading activity is a source of money. There are numerous markets were investments can be made. On these markets are traded: real goods like petrol or flowers, currencies, precious metals and shares. On these markets there are traders that use the goods for real, producers and consumers and traders that aim to use the goods as deposits of their money from which to obtain profit. On the markets, the prices are formed freely by demand and supply. There is a lot of research performed to analyze the evolution of prices from various markets. The basic rule of investment is to buy when the price is low and sell when the price is high. The analysis performed takes the forms:

- Technical analysis: where only quantitative methods, mathematical and statistical principles are applied to decide when to buy and when to sell
- Fundamental analysis, which is a qualitative approach and aims to identify the external factors that determine the evolution of the market.

However, the quantitative approach is more objective than the qualitative approach and offers the possibility of applying various scientific techniques. It offers a large domain for research and practically has concrete measurable results to assess the performance of a certain method.

2 Trading rules and genetic algorithms

The use of genetic algorithms is found in scientific literature to be used for deriving trading rules. In [1] a genetic algorithm approach for trading rules is presented. Genetic programming is used to model the market and also statistical methods are presented at large. However the emphasis is on modeling the behavior of traders.

In [2], also the set of traders of the market is modeled as a whole the trading rules being pre-established.

In [3], training rules of a fixed form taking into account the price of a security and the relative magnitude of the increase or decrease of the price. If the price moves up at least a parameterized percent, the trader buys and holds the security until the price moves down at least a parameterized percent. The rule is fixed structure and genetic algorithms are used to optimize parameters.

Genetic programming is presented in [4] along with methods to derive buy-sell rules. Complex expressions are built using a variety of operands and operators.

A comprehensive approach of generating trading rules using genetic programming cited by many researchers is found in paper [5]. Rules are generated on an if-then-else skeleton using arithmetic, relational and logical operators.
Genetic programming used in the above approaches is a particular type of genetic algorithm that not entirely follows the evolutionary principles as genetic operations apply directly to syntax trees not on linear structures like chromosomes. This happens due to the various flavors of operators that take different numbers of arguments and have different return types. This variety generates numerous restrictions when applying crossover and mutation like operations, as syntax trees must be kept valid after transformations are made. A criticism of genetic programming is found in [6].
In this paper gene expression programming is used to overcome drawbacks of genetic programming when deriving trading rules.
Paper [7] uses gene expression programming for model generation. Homogenous expressions, in terms of operands and operators types are evolved to estimate certain phenomena. In this approach only real operands were used along with operators that take real arguments and return real values.

3 Gene expression programming algorithm design

In order to apply a genetic type algorithm to a real life problem, an analysis phase is necessary in order to establish how real life elements map to abstract elements in evolutionary algorithms.
The chromosome is an abstract encoding of the characteristics that make up a modeled individual. If the individual is a neural network, then the chromosome encodes characteristics like weights and thresholds. If the individual is an estimation model then the chromosome encodes the expression of the model.
The gene is a component of a chromosome that encodes homogenous characteristics within the model. When the chromosome corresponds to a neural network, a gene inside a chromosome may encode a certain layer. In the case of a model, a gene may encode a sub-expression or a branch.
Chromosomes are made up of one or more genes. Chromosomes with more than one gene are called multigenic.
The symbol is the basic element that corresponds to the simplest part of the modeled element. In a chromosome modeling a neural network a symbol corresponds to weight or a threshold in the network. In a chromosome encoding an estimation model, the symbol encodes an operand or an operator.
When modeling trading rules there must be established the correspondence between the studied domain and the genetic structures.
The hypothesis has to be made that trading rules are buy or sell Boolean expressions made up of Boolean operands and Boolean operators. Taking into account the available data about exchange rates, and the fact that quotations are real numbers, further analysis has to be made to divide the problem into smaller problems that can be easily solved. The mapping between the concepts and the genetic structures is given in Table 1.

Table 1. Correspondence between concepts and genetic structures

Genetic structure	**Trading concept**
Chromosome	Trading rule
Gene	Expression
Symbol	Operand or Operator

When dealing with expression generation and all the operators are of the same type and operators are all of the same type and also fit to be passed as arguments to the operators, genes encode sub-expressions.
In a simple scenario, in order to obtain the individual, sub-expressions encoded by the genes may be simply aggregated by a linking

function. A more advanced handling uses a extra special gene made up of operators used to link the other genes. The gene G0 from Table 2 is the linking gene and the whole chromosome encodes the expression (a+b) * (c-d) / (f+g).

Table 2. Expression encoded by multigenic chromosome

Gene	G0	G1	G2	G3
symbols	*/	+ab	-cd	+fg

These extra genes may also take part in the evolutionary process and evolve on the same rules as the other genes.
A more general scenario is obtained when the encoded expressions are made up of operands and operators of different types.
in this scenario, genes are encoders of homogenous expressions. The operators of each gene type are homogenous, meaning they all take arguments of the same type as inputs and they all return the same type. The operators of each gene type are of the same type as the arguments the operators take. They may be simple symbols of required types or references to other genes where the aggregation operator also returns a compatible type.
This involves that each gene type to have a context which defines:

- the head size;
- the tail size; this is dependent on the head size and the maximum number of arguments the type of operators specific to the genes takes; this varies with the type of gene and has to be treated in the same context;
- the set of operators; the operators are homogenous inside the same type of gene
- the set of operands; it consists of multiple sets; the set of constants contains constants of the same type as the arguments; the set of variables contains variables from the dataset that are compatible with the required type; in the operand set there is also found the set of gene references that point to genes where operators return the compatible type.

Table 3. Multigenic chromosome with linking function gene

G0	G1	G2	G3	G4	G5	G6
|| && G1 G2 G3	<a G6	>G4 G5	>b c	+d f	-g h	*ij

The multigenic chromosome in table 3 shows an expression containing:

- in linking gene G0 – operators that take Boolean arguments and return Boolean values
- in genes G1, G2 and G3 – operators that take real arguments and return Boolean values
- in genes G4, G5, G6 – operators that take real arguments and return real values

The encoded expression is:
(a<G3)||(G4>G5)&&(b>c)
which furtherer is expanded by dereferencing into
(a<(i*j)) || ((d+f)>(g-h)) && (b>c)
which is valid regarding the compatibility of types.
Judging by the above three contexts are identified

- the context C1 of operators taking Boolean arguments and returning Boolean values; Operators={ ||, && } Operands ={ G1, G2, G3}
- the context C2 of operators taking real arguments and returning Boolean values; Operators = { <,>}, Operands = {a, b, c, d, f, g, h, i, j, G4, G5, G6}
- the context C3 of operators taking real arguments and returning real values; Operators = { +, -} Operands={ a, b, c, d, e, f, g}

This approach permits a linear representation of complex expressions, the chromosome being made up of genes in a certain order, given by the designer of the algorithm. Given a certain order of genes, crossover operations will always result in valid chromosomes ready to be evaluated.

Given the contexts above, the structure of the chromosome is shown in Table 4.

Table 4. Multigenic chromosome with multiple gene contexts

Context	C2	C2	C2	C2	C3	C3	C3
Gene	G0	G1	G2	G3	G4	G5	G6

Given the multigenic chromosomes shown in table 5 made up of genes corresponding to the contexts where c_i denotes a gene of context *i*, crossover at gene 2 leads to the new chromosomes shown in Table 6, which are chromosomes of the same gene structure.

Table 5. Multigenic chromosomes before cross-over

c_1’	c_2’	c_2’	c_2’	c_3’	c_3’	c_3’
c_1’’	c_2’’	c_2’’	c_2’’	c_3’’	c_3’’	c_3’’

Table 6. Multigenic chromosomes after crossover

c_1’	c_2’	c_2’’	c_2’’	c_3’’	c_3’’	c_3’’
c_1’’	c_2’’	c_2’	c_2’	c_3’	c_3’	c_3’

Also the validity maintains when the crossover point is at a random position that falls inside a gene, due to the basic properties of the gene expression programming chromosomes. Given the genes G_i and G_{i+1} of the same context C3, viewed at a closer level as shown in Table 7 crossover at position 34 leads to the situation presented in table 8, which further conserves the validity of the chromosome. Note that genes G_i and G_{i+1} have different head sizes.

Table 7. Genes from a chromosomes involved in cross-over at arbitrary position before the operation

Gene	Gi			Gi+1				
Position	30	31	32	33	34	35	36	37
Symbol	-	b	c	+	*	n	m	p
Gene	Gi			Gi+1				
Position	30	31	32	33	34	35	36	37
Symbol	*	f	g	/	-	h	i	j

Table 8. Genes from chromosomes involved in cross-over at an arbitrary position after the operation

Gene	Gi			Gi+1				
Position	30	31	32	33	34	35	36	37
Symbol	-	b	c	+	-	h	i	j
Gene	Gi			Gi+1				
Position	30	31	32	33	34	35	36	37
Symbol	*	f	g	/	*	n	m	p

Mutation operations must take into account the peculiar context the gene containing the position in the chromosome where the mutation takes place. The exchanged symbols must come from the corresponding sets of operands and operators.

Given the gene from Table 9, mutation at position 33, being located in the head of the gene must change the symbol but keep it of type operator. Changing the symbol 34 to “+”

gives the result presented in Table 10.

Table 9. Gene before suffering a mutation

Gene	Gi			Gi+1				
Position	30	31	32	33	34	35	36	37
Symbol	*	f	G	/	*	N	m	p

Table 10. Gene after suffering a mutation

Gene	Gi			Gi+1				
Position	30	31	32	33	34	35	36	37
Symbol	*	f	G	/	+	n	m	p

By these examples the idea of using gene expression programming along with linearization of expressions by means of homogenous context genes is highlighted.

Coding two elements inside the same chromosome is achieved by creating disjunctive chains of references. In the case study, both buy and sell rule were encoded inside a chromosome. Gene G0 encodes the expression for the buy rule; gene G1 encodes the expression for the sell rule. From gene G0 there are references to genes that encode sub-expressions of homogenous types, that themselves contain references to other genes. The same goes for gene G1 but the chain of references from G0 to the leaf level and the chain of references from G1 to the leaf level do not have common genes. This has impact in the evaluation of chromosomes when for a row of input data, containing he values of the variables participating in expressions, the evaluation is done either for the buy or for the sell. If the buying rule is verified, then the evaluator uses the expression in the chain of genes starting with G0; if the evaluator checks the sell rule, it will evaluate the expression encoded in the chain starting with G1 gene. The evaluation stage is particular to the model considered for the problem.

4 Case study

In the case study performed, rules of investment have been derived. The dataset used contains 1475 records of EUR-RON daily exchange rates from the National Bank of Romania, along with the derived indicators of simple moving average for 5 days, 10 days, 20 days, 50 days and 100 days, denoted by, respectively y, MA5, MA10, MA20, MA50 and MA100. Only the spot price and the moving averages were chosen, because the data comes from a public data source which does not provide information at a higher level of detail than daily values. In order to compute other technical indicators, more data would have been needed, like opening price, closing price and inner evolution during the transaction time span. The evolution of the daily EUR-RON exchange rate for the studied period is presented in Figure 1.

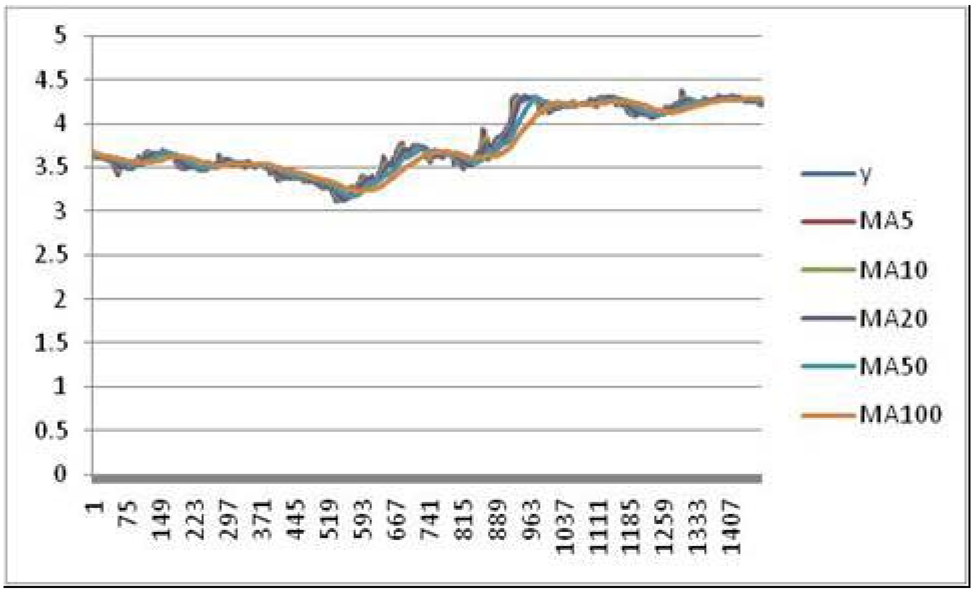

Fig. 1. The EUR-RON exchange rate for 1475 days

Commonly used trading rules state principles like:

- buy when the quotes climb above a conveniently chosen moving average; sell when the opposite happens; e.g. buy when y>MA50, sell when y<MA50;
- buy when moving average from *n* days climbs above the moving average from 2**n* days; sell when the opposite; e.g. buy when MA10>MA20, sell when MA10<MA20.

These common rules have a general nature and may not be used as they are on any dataset. Gene expression programming in the form presented in the previous chapter is used do find proper rules for the buying and selling signals for this dataset.

The hypotheses used in the case study are:

- the investor has an initial sum of 1000 RON for investment;
- the initial state of the investor is BUYING; this means the investor waits a buying signal like a rule that applies; the rule is described by a Boolean expression that is assessed for each day;
- if the Boolean expression that describes the buying rule is true, the investor uses all the money to buy the EUR currency; the investor changes its state to SELLING; no more buying occurs as all the money in RON have already been used to buy the currency;
- in SELLING state the investor waits for a sell signal; this comes when the selling rule described by a Boolean expression becomes true; when selling all the money in the foreign currency are transformed into RON;
- all the transactions are made **after the signals are received**, in the first moment the quotation is favorable, meaning selling when the quotation is greater than last time when buying, and buying when the quotation is less than the last time when selling;
- at the end of the investment period the last sum earned in RON is considered and compared against the initial sum of 1000 RON.

The setup of the genetic algorithm uses three main gene contexts, customized per chromosome part. The chromosome encodes both buy and sells rules.

For the buying rules, the gene contexts are:

- **bbcontext1**- the context of operators taking Boolean operators and returning Boolean values : Operators = { ||, &&, @}, where @ is an identity operator, Operands = {G2,G3};
- **dbcontext1** – the context of operators taking double arguments and returning Boolean values: Operators = { <,>}, Operands = { y, MA5, MA10, MA20, MA50, MA100, G6,G7};

- **ddcontext1** – the context of operators taking double arguments and returning double values; Operators = {-}, Operands={ y, MA5, MA10, MA20, MA50, MA100}.

For the selling rules, the gene contexts are:

- **bbcontext2**- the context of operators taking Boolean operators and returning Boolean values : Operators = { ||, &&, @}, where @ is an identity operator, Operands = {G4,G5};
- **dbcontext2** – the context of operators taking double arguments and returning Boolean values: Operators = { <,>}, Operands = { y, MA5, MA10, MA20, MA50, MA100, G8,G9};
- **ddcontext2** – the context of operators taking double arguments and returning double values; Operators = {-}, Operands={ y, MA5, MA10, MA20, MA50, MA100}.

The structure of the chromosome is {G0, G1, G2, G3, G4, G5, G6, G7, G8, G9}. The buying rule is encoded by the genes {G0, G2, G3, G6, G7}. The selling rule is encoded by {G1, G4, G5, G8, G9}. Genes G0, G1 belong to a **bbcontext**, genes G2, G3, G4, G5 belong to a **dbcontext**, genes G6, G7, G8 and G9 belong to a **ddcontext**.
The chromosome,

[@ G3 G3][@ G5 G5][< G7 G6][< G6 G7][<G9 G8][< G8 G9][- MA5 MA10][- MA20 y][- MA10 y][- MA5 MA20]
encodes the rules
BUY :(MA5-MA10<MA20-y)
SELL :(MA10-y<MA5-MA20)

The dataset has been divided into two series: a training series containing a proportion of 0.6 of the records, used for deriving the rules and a testing series containing a proportion of 0.4 of the records, used for assessing the performance of the derived rules. After subsequent runs of the algorithm several rules have been generated and chosen as seen in Table 11.

Table 11. Trading rules evolved via the gene expression programming

Rule id	Raw output
R1	[@ G3 G3][@ G5 G5][> MA5 MA20][> MA5 MA100][> MA100 MA50] [< MA5 MA100][- MA20 MA50][- y MA100][- y y][- MA100 MA100] BUY :(MA5>MA100) SELL:(MA5<MA100) Training performance:1209.70 Testing performance: 1148.84
R2	[@ G2 G3] [&& G4 G4] [< MA100 MA10][< G6 MA10][< MA10MA100] [> MA20MA100][- MA10 y][- MA50 MA10][- MA10 MA10][- MA10 y] BUY :(MA100<MA10) SELL:((MA10<MA100)&&(MA10<MA100)) Training performance:1303.02 Testing performance: 1109.20
R3	[@ G3 G2][@ G5 G4][> MA10 MA10][> MA20 MA50][> MA10 MA50] [< MA20 MA50][- MA100 MA100][- y MA20][- MA10 MA10][- MA50 MA20] BUY :(MA20>MA50) SELL:(MA20<MA50) Training performance:1329.55

	Testing performance:1170.27
R4	[@ G2 G2][@ G4 G5][< y MA20][> MA50 G7][> y MA20] [> MA50 G9][- MA20 MA100][- y MA50][- MA10 MA5][- MA5 MA100] BUY :(y<MA20) SELL :(y>MA20) Training performance:1132.14 Testing performance:1068.60
R5	[&& G2 G2] [@ G4 G4] [> MA5 MA50] [> y G6] [< MA5 MA50] [> MA10 G8] [- y MA5] [- MA50MA100] [- MA10 MA5] [-MA100 MA50] has Training performance:1079.39 Testing performance:1196.78 BUY :((MA5>MA50)&&(MA5>MA50)) SELL:(MA5<MA50)
R6	BUY :(MA20-MA50>MA50-MA10) SELL :(MA20-MA50<MA50-MA10) It took -320414 Training performance:1345.60 Testing performance:1223.26

For the rule R3:

BUY :(MA20>MA50)
SELL:(MA20<MA50)

```
the sequence of operations from the
testing period is
Buying when y is 3.7271
Selling when y is 3.7835
Buying when y is 3.7794
Selling when y is 4.2348
Buying when y is 4.1977
Selling when y is 4.2216
Buying when y is 4.2157
Selling when y is 4.2249
Buying when y is 4.1484
Selling when y is 4.2347.
```

It is observed that buying and selling occur at favorable moments avoiding losses.
From the list of raw outputs it is seen that for the considered dataset, many of the generated rules take into account moving averages that are in the relation {n , more than two times n} as is {n, 10*n}, e.g. buy when y>MA10, sell when y<MA10, buy when MA10>MA100, sell when MA10<MA100, buy when MA5>MA50, sell when MA5<MA50. This is specific to this dataset and is a rule detected by the evolutionary algorithm.
It is observed that for rule R6,
BUY :(MA20-MA50>MA50-MA10)
SELL :(MA20-MA50<MA50-MA10)
a more complex expression has been obtained through evolution. The rule is rewritten as
BUY :(MA20 > MA50 + (MA50 - MA10))
SELL:(MA20 < MA50 + (MA50 - MA10))
and it is observed that the evolution of the rules tries to obtain other values based on the existing values of the variables, in a manner similar to the estimation of unknown parameters through evolving arithmetical expressions between initial random constants. The generated rules along with the hypotheses made avoid losses and give profit to the trader.

5 Conclusions

Useful trading rules are derived using genetic algorithms, in general and gene expression programming in particular. Gene expression programming is a very powerful method that allows solving of very complex problems that require generating expressions. Gene expression programming follows very closely the principles of genetic evolution.

The design of the solution using a genetic algorithm is necessary. This affects the implementation of the algorithm. Also the flexibility of applying a certain solution to other similar problems must be taken into account.
The generated rules are according to the practices in the domain, but they customize the behavior of the trader to the peculiarities of the dataset being considered.

References

[1] S. Heng Chen, *Genetic algorithms and genetic programming in computational finance*, Vol. 1, Springer, 2002.
[2] C. Lawrenz and F. Westerhoff, "Modeling Exchange Rate Behavior with a Genetic Algorithm", *Computational Economics*, Vol. 21, No. 3, pp. 209-229.
[3] L. Lin, L.B. Cao, J.Q. Wang and C.Q. Zhang, "The applications of genetic algorithms in stock market data mining optimization," In *Proc The Fifth International Conference on Data Mining, Text Mining and Their Business Applications*, Malaga, Spain, 2004, pp. 273-280
[4] J. Y. Potvin, P. Soriano and M Valee, "Generating trading rules on the stock markets with genetic programming," *Computers and Operations Research*, No 31, 2004, pp. 1033-1034.
[5] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Financial Economics*, No. 51, 1999.
[6] C. Ferreira, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence 2nd Edition*, Springer Publishing, May 2006.
[7] A. Visoiu, "Structure Refinement for Vulnerability Estimation Models using Genetic Algorithm Based Model Generators," *Informatica Economica Journal*, Vol. 13, No. 1, 2009.

Adrian VIȘOIU graduated the Bucharest Academy of Economic Studies, the Faculty of Cybernetics, Statistics and Economic Informatics. He has a master degree in Project Management. He has a PhD in the field of software quality. He is a Software Engineer in telecom field and a collaborator of Economic Informatics Department of the Bucharest Academy of Economic Studies. He published articles alone or in collaboration and he is coauthor of three books. His interests include: programming, genetic algorithms and neural networks.