Enhanced Gene Expression Programming for signal-background discrimination in particle physics

Liliana Teodorescu[*]
Brunel University
E-mail: `Liliana Teodorescu@brunel.ac.uk`

Zhengwen Huang
Brunel University
E-mail: `Zhengwen.Huang@brunel.ac.uk`

The original version of Gene Expression Programming, a variant of Evolutionary Algorithms, was enhanced in this study with an alternative representation of the candidate solution based on a prefix notation, and with a truncated evolution mechanism. The algorithm was applied to a signal-background classification problem for which a dynamic classification threshold was implemented. As an example application the selection of K_S particles produced in e^+e^- interactions at 10 GeV and reconstructed in the decay mode $K_S \to \pi^+\pi^-$ was used. All these developments resulted in an algorithm more efficient in terms of its convergence speed, measured in number of generations, as well as in slight improvements of the quality of the final solution measured in terms of the classification accuracy.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7 2008
Erice, Italy

[*]Speaker.

1. Introduction

Gene Expression Programming (GEP) is a relatively new version of Evolutionary Algorithms developed in 2001 [1]. It combines and extends some of the ideas implemented in the more established versions, Genetic Algorithms [2] and Genetic Programming [3], resulting in a variant which overcomes some of the individual limitations of these versions.

The common approach of all Evolutionary Algorithms is to find the solution to a problem by iteratively improving the quality of a candidate solution through a process which simulates the natural evolution. The candidate solution is encoded into a form understood by the computer called a chromosome. The candidate solution is changed through a process called genetic variation achieved by applying a set of operators, called genetic operators, on the chromosome. The quality of the candidate solution is assessed with an objective function called fitness function.

The main difference among different Evolutionary Algorithms variants is the way the candidate solution is encoded (represented) and, consequently, the structure of the genetic operators which need to take into account the specific representation method. GEP uses two entities to represent the candidate solution, a chromosome, which is the encoder of the candidate solution, and a tree, called an expression tree, obtained through a translation process from the chromosome. The expression tree corresponds to a mathematical expression which represents the actual candidate solution to the problem.

The applicability of GEP to particle physics data analysis was investigated in our previous work [4] [5] [6] for an event selection (signal-background discrimination) problem. This paper presents a continuation of those studies by investigating a series of techniques to improve the effectiveness of the algorithm.

2. Gene Expression Programming

The full description of GEP is presented in [7] and summarised in [4] and [5]. Only a brief summary is presented here.

The candidate solution to the problem at hand is encoded into a chromosome which is a list of functions and terminals (variables and constants) organised in segments of equal length called genes. Each gene has a "head" made of functions and terminals, and a "tail" made only of terminals. The length of the head (h) is an input parameter to the algorithm, while the length of the tail is given by $t = h(n-1)+1$, where n is the largest arity of the functions used in the gene's head.

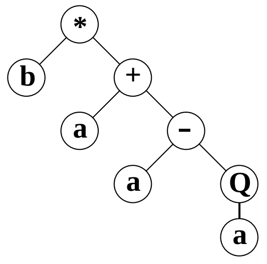
Each gene of the chromosome is decoded into an expression tree (ET). An example of a chromosome made of one gene, the decoded ET and its corresponding mathematical expression is presented in Figure 1. The decoding process implies placing the first element of the gene on the first line of ET and then, on each next line, placing as many elements as required by the functions of the previous line. The process continues until all elements of the ET's line are only terminals. Depending of the actual composition of the chromosome, the decoding process can end before the end of the gene (even before the end of the gene's head). This means the chromosome can have non-expressed regions (just like the biological genes) which contain additional information used during genetic variation whenever needed in order to create syntactically correct structures (ET's).

Chromosome with one gene

*b+a-aQab+//+b+babbabbbababbaaa

head | tail

Expression tree



Mathematical expression

$$b * (a + (a - \sqrt{a})) = b * (2a - \sqrt{a})$$

Figure 1: Unigenic chromosome, the decoded ET and its corresponding mathematical expression (Q- square root function; a,b - terminals)

In the case of multigenic chromosomes, the ET's corresponding to each gene are connected with a linking function defined by the user. The mathematical expression associated with these combined ET's is the candidate solution to the problem.

An initial population of chromosomes is created by randomly selecting functions and terminals from the set chosen by the user. Then each chromosome is decoded into an ET which represents the mathematical expression corresponding to a candidate solution. The quality of the candidate solution (chromosome) is evaluated using a fitness function defined by the user and adapted to the problem to be solved. The value evaluated for this function is called chromosome's fitness.

After the fitness of all individuals in the population was calculated, a termination criterion is evaluated. This criterion is, usually, a level of quality of the candidate solution or a maximum number of generations created. If the termination criterion is not met, a set of chromosomes are selected with a probability proportional to their fitness and transformed by applying genetic operators on them. This process is called reproduction. The new individuals constitute a new population and the process is repeated until the termination criterion is met. The best chromosome in the final population is selected and decoded, resulting in the solution to the problem as found by the algorithm.

The genetic operators used in GEP are cross-over (recombination) which exchanges parts of a pair of chromosomes, mutation which randomly changes an element of a chromosome into another element, preserving the rule that the tail contains only terminals, and transposition which moves a part of the chromosome to another location in the same chromosome. Each operator is applied with a certain probability, called operator rate, which needs to be optimised for the problem studied.

3. Problem studied and datasets

Using a statistical learning approach, GEP was used to extract selection criteria for a signal-background classification problem with the purpose of investigating the behaviour and performance of the algorithm, rather than to extract a physics result.

In order to be able to rely on the conclusions of the previous studies ([4] [5] [6]) and to perform meaningful comparisons, the same datasets as in these previous studies were used. They are Monte Carlo data corresponding to the decay process $K_S \to \pi^+\pi^-$ with K_S produced in e^+e^- interaction at $\approx$10 GeV in the BaBar experiment [8]. The e^+e^- interaction was simulated generating $e^+e^- \to q\bar{q}$ events (q being a quark and $\bar{q}$ an antiquark) with JETSET [9] (for q being the u, d, s and c quarks) and EvtGen [10] (for q being the b quark) simulation packages. The generated events were passed through the detector response simulation package [11] and reconstructed with the BaBar offline analysis software. Signal events were defined as those containing a reconstructed K_S particle associated with a generated K_S particle and for which the reconstructed π daughters were associated with the π daughters of the generated K_S particles. All the other reconstructed events were defined as background.

Training and test data samples of equal sizes (5,000 events) with a signal-to-background ratio of 1:4 were used. Each dataset contained 8 event variables as input to the analysis. They were variables usually used in a standard cut-based analysis for the K_S decay process. The full list and explanations can be found in [5]. It was previously demonstrated [6] that, for this problem, the increase of the number of events or of the number of event variables in the dataset do not improve the quality of the solution. The algorithm has the capability to select automatically the relevant event variables and no overtraining was observed with the increase of the number of events.

The constants used in building the solution were created by the algorithm itself in the range (-10,+10). The boundaries of the range were given as input to the algorithm. Selection rules were extracted from the training data samples and tested on the corresponding test data samples.

The set of input functions from which the algorithm builds the chromosomes contained 36 common mathematical functions [5]. In this study it was also shown that, by restricting the set of input functions to common logical functions, the algorithm finds automatically selection rules similar to the cut-based rules used in a standard cut-based data analysis for the physics process studied (in which the selection cuts are chosen based on physics considerations and their values optimised manually), proving the algorithm works correctly.

Other GEP input parameters, found optimal for this problem in the previous studies , were: the length of the gene head equal to 10, the number of chromosomes per generation equal to 100, and the maximum number of generations equal to 20,000. The genetic operator rates were kept as recommended in [7]: 0.044 for mutation, 0.3 for transposition and 0.1 for recombination.

The fitness function was the number of hits (the number of events correctly classified as signal or background).

The performance of the algorithm was analysed in terms of the classification accuracy defined as the ratio of the total number of events correctly classified (signal and background) to the total number of events of the sample.

In this study a private software implementation of the algorithm was used. As part of its validation, extensive comparisons of its results with those obtained with GeneXproTools [12], the software package created by the algorithm developer and used in our previous studies, were performed. The relative difference of the classification accuracies obtained with the two software implementations was less than 0.1% in all cases.

4. Enhancements of the algorithm

A number of techniques to improve the quality of the solution and the efficiency of the algorithm were investigated in this study. In order to average the statistical fluctuations due to the stochastic character of the algorithm, each version of the algorithm was run ten times in identical conditions (identical input information) and the average classification accuracy was calculated and used in comparison studies.

The difference between the results obtained with different versions of the algorithm was statistically analysed with the Students' t test method [13] and its significance level calculated and reported.

4.1 Prefix decoding

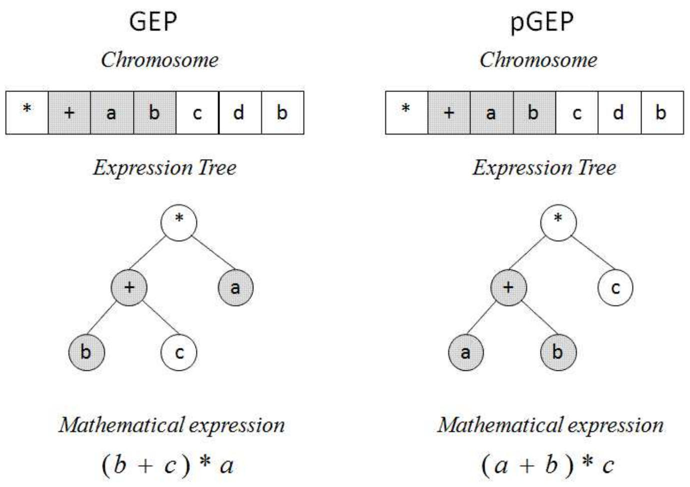


Figure 2: Example of the GEP and pGEP decoding methods

The original GEP uses a width-first decoding method, described in section 2, for mapping the chromosome into an expression tree. In [14] a different method based on a prefix order notation was proposed. Figure 2 shows an example of decoding the same chromosome with the two methods. The prefix decoding starts with placing the first element of the chromosome on the first line of ET and, if it is a function, placing the second element on the next line, as its first argument. If the second element is also a function then the next elements of the chromosome are placed on the next line, as arguments of the function. The process continues, following this depth-first approach, until the entire branch is completed by ending with terminals. Then the next element of the chromosome is placed on the second line of ET, as the second element of the first function, and the process continues until ET is completed by ending all its branches with terminals. The GEP version based on the prefix decoding method is called pGEP in this study. It should not, however, be assimilated with the pGEP algorithm proposed in [14] which contains additional modifications of the original GEP algorithm not considered here. These additional modifications abandoned some of the novel ideas implemented in GEP, such as the head-tail separation of the chromosome, making a step backwards towards other previously proposed versions of evolutionary algorithms.

pGEP maintains better the proximity of the genetic material (chromosome's elements) during the translation process into ETs and this is expected to help the evolution process by reducing the destructive effect of the genetic operators. It is more likely that the related genetic material remain together in pGEP than in GEP during the evolution process.

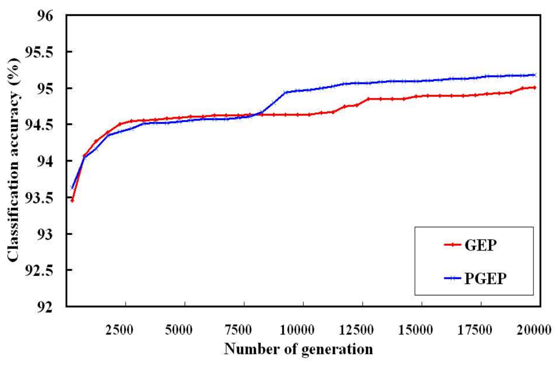


Figure 3: Classification accuracy as a function of the number of generations for GEP (red) and pGEP (blue)

The results obtained with GEP and pGEP for the problem studied here are presented in Figure 3 which shows the dependence of the classification accuracy obtained with the two algorithms (GEP in red and pGEP in blue) as a function of the number of generations. It can be seen that, indeed, pGEP if faster than GEP as it reaches the convergence sooner (around 10,000 generations). Also, the classification accuracy obtained with pGEP after 20,000 generations is slightly higher than that obtained with GEP. The two values are different at 35% significance level.

This result indicates that maintaining the proximity of the related genetic material during the evolution process has a positive effect on the efficiency of the algorithm. It also suggests as useful to investigate mechanisms to control this proximity during the evolution. Further studies in this direction will be performed.

4.2 Truncated evolution

Each generation, particularly in the early stage of the evolution process, is expected to have a number of individuals of low quality. In a normal evolution these individuals are fully processed (take part in the selection process) and have a certain probability to participate in the reproduction process. Truncated evolution is the evolution in which these low quality individuals are totally eliminated with the expectation that this will improve the efficiency of the search process.

In this study the truncated evolution was implemented by imposing a certain fitness threshold (FT). Only individuals with the fitness value higher than FT were allowed to participate in the reproduction process. Imposing such a threshold has two effects which need to be balanced. On one hand, it will improve the convergence speed (number of generations in which the solution is found). On the other hand, it facilitates the reduction of the population diversity which might favour the trapping of the algorithm in a local optimum. The value of FT has to be carefully optimised in order to balance the two effects.

The FT used was guided by the average fitness value per population and it was called an online fitness threshold. It was calculated by multiplying the average fitness per population with a scaling

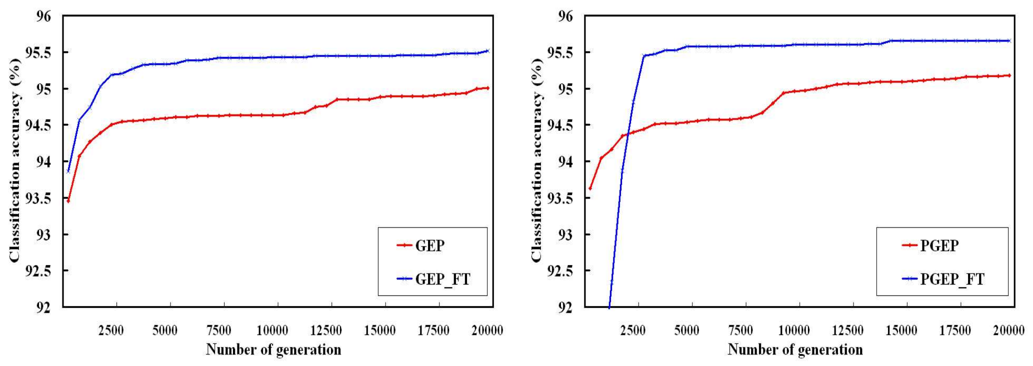


Figure 4: Classification accuracy as a function of the number of generations for GEP (left) and pGEP (right) with normal evolution (red) and with truncated evolution (blue)

factor which needs to be optimised for each problem. Typical values of the scaling factor were between 0.5 and 1.5.

This online FT was found to provide a better pressure on the evolution process, if it is properly optimised. If the value of FT is too high then unstable results are obtained due to a high degree of uniformity of the population resulting in trapping the algorithm in local optima.

The results obtained by imposing an online FT are presented in Figure 4 for GEP (left) and pGEP (right). The optimal scaling factor was found equal to 1.25 in both cases. It can be seen that this method produces an earlier convergence (under 5,000 generations), particularly for pGEP, as well as an improvement of the classification accuracy at the end of the search process. The values obtained with and without truncated evolution are different at less than 1% significance level for both GEP and PGEP.

4.3 Dynamic classification threshold

Most methods for signal-background classification produce a continuous output (function) on which a classification threshold is applied in order to define the signal and background events, as separated by the algorithm. This threshold is chosen at the end of the search process and adapted to the final output. This approach is not adequate for methods based on Evolutionary Algorithms as each individual (candidate solution) provides its own output which has its own optimum classification threshold.

In order to address this problem the optimal classification threshold was searched for each individual by scanning the full range of the output function. This classification threshold was called a dynamic classification threshold as it changes its value during the evolution process.

The results obtained with this method are presented in Figure 5 for GEP (left) and pGEP (right). It can be seen that this methods provides a faster convergence in terms of number of generations, particularly for pGEP (not much for GEP), as well as slightly higher classification accuracies at the end of the process. The final classification accuracies obtained with and without dynamic classification thresholds are different at 20% and 12% significance levels for GEP and PGEP, respectively.

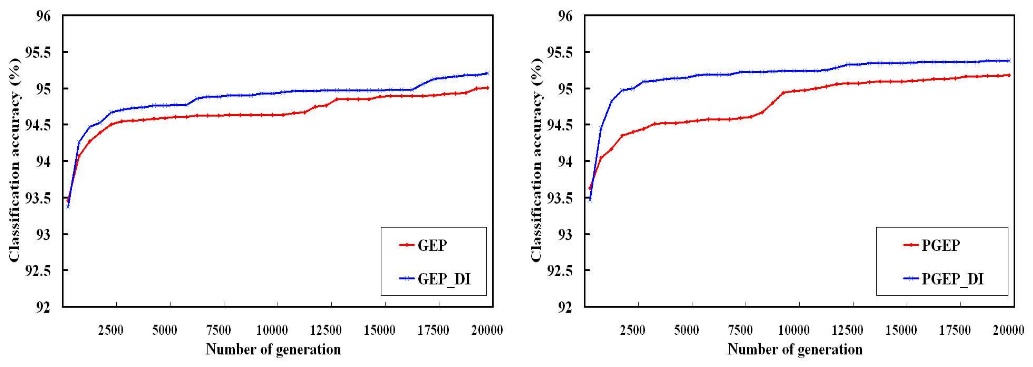


Figure 5: Classification accuracy as a function of the number of generations for GEP (left) and pGEP (right) with constant (red) and dynamic (blue) classification threshold

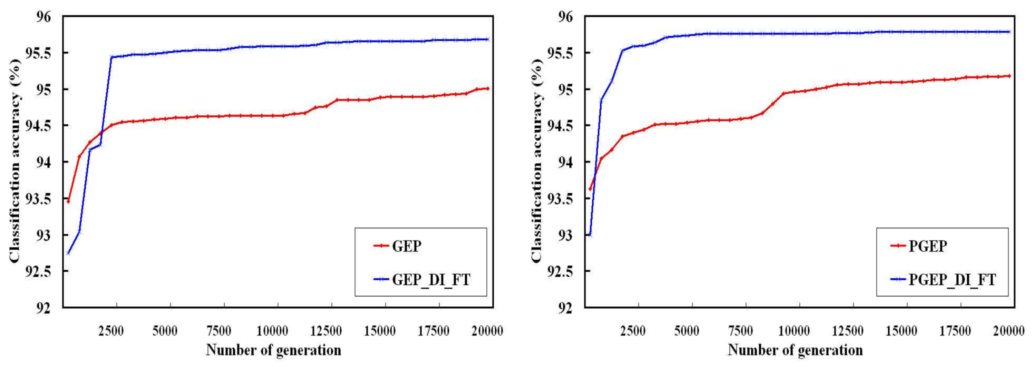


Figure 6: Classification accuracy as a function of the number of generations for GEP (left) and pGEP (right) with constant (red) and dynamic (blue) classification threshold and with truncated evolution

The dynamic classification threshold was also implemented together with truncated evolution and the results are presented in Figure 6 for GEP (left) and pGEP (right). The earlier convergence (under 5,000 generations) of the new versions of the algorithms can be seen also in this case, as well as improvements of the final classification accuracy. The classification accuracies with and without dynamic classification threshold and truncated evolution are different at less than 1% significance level for both GEP and pGEP.

4.4 Combined developments

Combining the developments described in the previous sections, the highest performance is obtained with the version of the algorithm using a prefix decoding, with truncated evolution and dynamic classification threshold. The comparison between this version and the original GEP is presented in Figure 7. The most significant improvement is in terms of the convergence speed, the number of generations needed to reach the optimal solution being under 5,000 generations. The quality of the final solution is also improved, the classification accuracy being with approximately 0.8% higher. The significance level of this difference is under 1%.

The improvement in the classification accuracy is not expected to be high for this problem as it is a relatively simple problem and the original GEP is expected to perform well from this point of

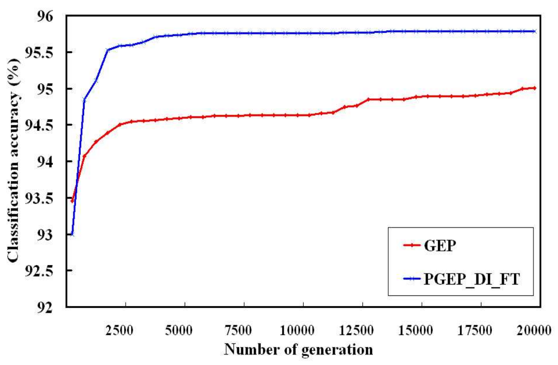


Figure 7: Classification accuracy as a function of number of generations for pGEP with dynamic classification thresholds and truncated evolution (blue) and the original GEP (red).

view. This can be seen also from the fact that the starting point of the search correspond to a high classification accuracy, around 93% (the value at the first generation). The trend, however, might be important for more complex problems and this aspect will be investigated in further studies.

While the convergence speed in terms of the number of generations is faster for the new version of the algorithm, the total running time is approximately the same for both versions as considerable more running time is needed per generation in order to create individuals over the fitness threshold and to optimise the classification threshold for each individual. Further studies to reduce this running time will be performed.

4.5 Generalisation power

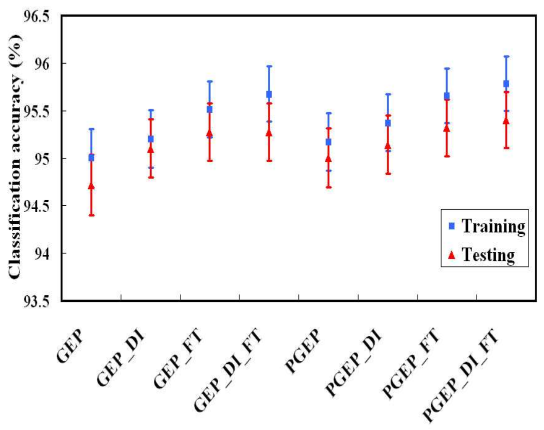


Figure 8: Classification accuracy on training (blue) and test (red) datasets for all versions of the algorithm.

The quality of the solutions developed by all versions of the algorithm was evaluated also in terms of their generalisation power. These results are summarised in Figure 8 which shows the classification accuracy on the training and test datasets for all versions of the algorithm, together with the corresponding statistical uncertainty due to the limited number of events in the data samples. It can be seen that the test classification accuracy follows closely the training classification accuracy, with the difference between the central values less than 0.5%, and that they are equal in the limit of the statistical uncertainty.

5. Conclusions

The original version of Gene Expression Programming was enhanced in this study with an alternative mapping method between the chromosome and the expression tree, and with a truncated evolution mechanism. Its application to a classification problem also implemented a dynamic classification threshold, meaning a mechanism for the optimisation of this threshold for each individual used in the search process.

The alternative chromosome - expression tree mapping implemented was based on a prefix notation approach which favours the maintenance of the proximity of the related genetic material during the evolution process. The improvement observed in the convergence speed, measured in terms on the number of generations, indicates the usefulness on keeping the related genetic material together during the evolution. Further studies to exploit this behaviour will be performed.

The truncated evolution implemented allowed an earlier selection of the good candidate solutions which resulted in an approximately 75% reduction of the number of generations needed to reach the convergence of the search process. It also allowed a statistically significant increase of the classification accuracy which defined the quality of the solution.

The dynamic classification threshold allowed a more accurate evaluation of the quality of each individual, as its fitness was dependent on the value of the classification threshold. This also contributed to the improvement of the performance of the algorithm in terms of both convergence speed (measured as the number of generations in which the solution is found) and classification accuracy.

The implementation of these mechanisms resulted in a version of the algorithm approximately 75% more efficient in terms of number of generations upto convergence and a slight improvement of the final classification accuracy, approximately 0.8% for the relatively simple problem investigated here, value which is statistically significant at a level lower than 1%.

These conclusions will be tested for more complex problems from particle physics in order to fully evaluate their potential. Also, mechanisms to reduce the running time of the enhanced algorithm will be investigated in order to allow a full exploitation of its beneficial behaviours observed in this study.

References

[1] C. Ferreira, *Gene Expression Programming: A New Adaptive Algorithm for Solving Problems*, Complex Systems, vol. 13, issue 2, pp. 87-129, 2001.

[2] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[3] J. R. Koza, *Genetic Programming: On the Programming of the Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

[4] L. Teodorescu, *Gene Expression Programming Approach to event selection in High Energy Physics*, IEEE Transaction on Nuclear Science, vol. 53, pp. 2221-2227, 2006.

[5] L.Teodorescu, D. Sherwood, *High Energy Physics event selection with Gene Expression Programming*, Computer Physics Communications 178, pp. 409-419, 2008.

[6] L. Teodorescu, I.D.Reid, *Application of Gene Expression Programming to Event Selection in High Energy Physics*, XI International Workshop on Advanced Computing and Analysis techniques in Physics Research, April 23-27 2007, Amsterdam, the Netherlands; Proceedings of Science 051, 2007.

[7] C. Ferreira, *Gene Expression Programming: Mathematical Modelling by an Artificial Intelligence*, Angra do Heroismo, Portugal, 2002.

[8] B. Aubert et.al., *The BaBar detector*, Nuclear Instruments and Methods in Physics Research, vol. A479, pp. 1-116, 2002.

[9] T. Sjostrand, *High-energy-physics event generation with PYTHIA 5.7 and JETSET 7.4*, Computer Physics Communications, vol. 82, pp. 74-89, 1994.

[10] D. Lange, *The EvtGen particle decay simulation package*, Nuclear Instruments and Methods in Physics Research, vol. A462, pp. 152-155, 2001.

[11] S. Agostinelli et. al., *GEANT4 - a simulation toolkit*, Nuclear Instruments and Methods in Physics Research, vol. A506, pp. 250-303, 2003.

[12] http://www.gepsoft.com

[13] see, for example, R. Barlow, *Statistics - A guide to the Use of Statistical Methods in the Physical sciences*, John Wiley & Sons, 1988.

[14] X. Li et. al, *Prefix Gene Expression Programming*, Genetic and Evolutionary Computation Conference (GECCO-2005), June 25-29 2005, Washington DV, USA.