Institute of Physics Publishing
doi:10.1088/1742-6596/32/1/010
Journal of Physics: Conference Series **32** (2006) 58–65
Sixth Edoardo Amaldi Conference on Gravitational Waves

Prospects of gravitational wave data mining and exploration via evolutionary computing

M. Lightman, J. Thurakal, J. Dwyer, R. Grossman, P. Kalmus, L. Matone, J. Rollins, S. Zairis, S. Márka

704 Pupin Laboratory, Physics Department Columbia University New York, NY 10027

Abstract. Techniques of evolutionary computing have proven useful for a diverse array of fields in science and engineering. Because of the expected low signal to noise ratio of LIGO data and incomplete knowledge of gravitational waveforms, evolutionary computing is an excellent candidate for LIGO data analysis studies. Using the evolutionary computing methods of genetic algorithms and genetic programming, we have developed, as a proof of principle, search algorithms that are effective at finding sine-gaussian signals hidden in noise while maintaining a small false alarm rate. Because we used realistic LIGO noise as a training ground, the algorithms we have evolved should be well suited to detecting signals in actual LIGO data, as well as in simulated noise. These algorithms have continuously improved during the five days of their evolution and are expected to improve further the more they are evolved. The top performing algorithms from generation 100 and 199 were benchmarked using gaussian white noise to illustrate their performance and the improvement over 100 generations.

1. Introduction

The development of gravitational wave[1] search algorithms for the Laser Interferometer Gravitational Wave Observatory (LIGO)[2] can be a complex task. The expected signals have a very low signal to noise ratio (SNR) with uncertain and varied waveforms. We have applied evolutionary computing (EC)[3] methods to demonstrate that EC is able to develop efficient data analysis algorithms. The primary motivation for using EC is that search algorithms can be trained to find signals in non-gaussian noise. We utilize two sub-fields of EC: genetic algorithms (GAs)[4] and genetic programming (GP)[5]. GAs use bit strings or continuous parameters to represent a solution, while GP uses more complex tree data structures for solutions. We have interlocked the two to create our program, providing an effective way to traverse the complicated solution space of gravitational wave search algorithms. GAs have been successful at solving complex problems more efficiently and within a shorter design cycle than their human counterparts[6].

To understand the evolution and to be able to decipher the result, we intentionally chose a simple model with a comprehensible solution space (see Section 2.3) and a basic fitness function (see Section 2.4). We first chose to examine coincident short burst signals, knowing that an evolutionary program developed for this purpose could also be trained on many different types of waveforms and even on multiple types simultaneously. The final product of this project is not the search algorithms themselves, but rather the environment that produces them. We found that the solutions created by the program have steadily advanced in their ability to find smaller signals more effectively while also decreasing their false alarm rate. These solutions have the potential to continue to improve as the evolutionary program's run duration increases. The result upon our arbitrary termination of the program was a few top search algorithms with similar efficiency. We performed benchmark tests on top solutions from midway through

the run and at the end of the run. Analysis of these search algorithms on signals in white noise shows that the algorithm at the end of the run outperforms the algorithm from an earlier generation, indicating that evolution has taken place. Receiver Operating Characteristic (ROC) curves are obtained for both of these algorithms, which indicate that even these early results are promising.

2. Design of the Evolutionary Program

The first step in an evolutionary algorithm[6, 7] is random generation of an initial population of potential solutions. These solutions are called *chromosomes*. Each chromosome is then evaluated by a user-defined *fitness function* and given a fitness score representing its quality as a solution to the problem. The fittest chromosomes are chosen as parents to create new chromosomes in an operation called *crossover*[8]. The parents and offspring are then *mutated*[9] by randomly changing specific values inside the chromosome. Their fitness is then recomputed. The least fit of the population are eliminated until the population is returned to its initial size. This basic evolutionary cycle, beginning with parent selection, is called a *generation*. The cycle repeats until a certain number of generations have been completed or a member of the population achieves a certain level of fitness.

2.1. Structure of the Chromosomes

In our algorithm a chromosome takes coincident data streams from two detectors. The use of two detectors was an arbitrary choice, as we do not foresee any limitation to stop us from using n data streams in future runs. A chromosome's ability to find signals is used to determine its fitness.

The basic structure of each chromosome is a binary parse tree which represents an evolving algorithm used to determine whether or not a given data set contains a signal. Nodes in the tree correspond to logical operators - 'and' or 'or'- and leaves correspond to questions the algorithm asks about the data[10]. An answer to these questions (which are described in detail below in Section 2.3) is either 'true' or 'false', and if the parse tree evaluates to 'true' overall, a signal should be present. An example of an actual parse tree produced by our program is shown in Figure 1.

Each chromosome in the population also requires its own vector of parameters which are needed to define the questions posed by the chromosome's algorithm. These parameters (also explained below in Section 2.3) evolve along with the algorithm itself.

2.2. Crossover and Mutation

Crossover is performed on the parameter vector and the binary tree. In a population of N chromosomes $N/2$ parents are selected to create $N/2$ offspring. For the parameter vector a GA technique known as *whole arithmetic recombination*[3] is used. We achieve the crossover of two parent trees by first selecting a random crossover point to represent the top of a sub-tree for each parent. The operation is then completed by swapping the sub-trees of the two parents. This creates two new trees, one for each offspring.

We also perform mutation on both the parameters and the binary trees. When a parameter is mutated its value is changed to a random value within its allowed range. We achieve tree mutation by randomly selecting a spot (or spots) on the tree and either randomly change the logical operator or question at that spot, or insert an entirely new, random sub-tree at that spot. GAs have a proven niche in multivariate optimization problems in large part due to mutation-based parameter tuning[4].

2.3. Questions in the Binary Tree

In order to analyze the data, each chromosome can ask various questions for which the answer is either 'true' or 'false'. The answer to a question is based on statistics calculated on the 1 second of data to which the question is restricted. For the purposes of this proof-of-concept experiment, we have chosen four fundamental question types. In the future, other types of questions might also prove useful to increase the size of the solution space being searched.

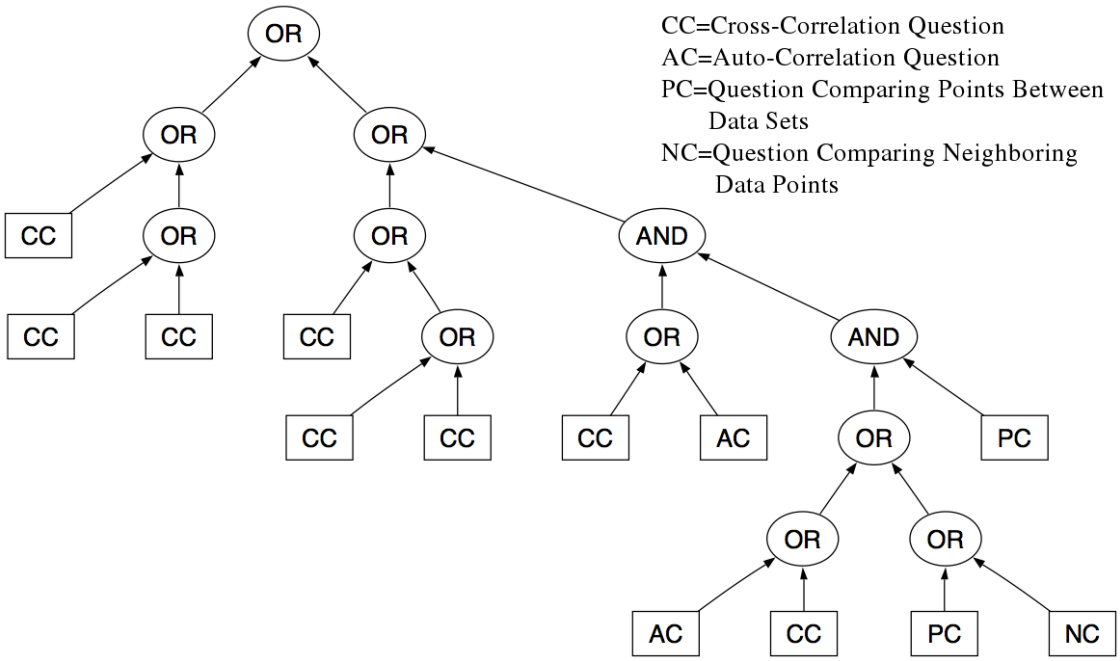

Figure 1. Our signal detecting algorithms are represented as binary tree logic systems that decide whether or not a signal is contained in the data. This particular tree is associated with one of the best algorithms evolved by our program: a top chromosome in generation 199. (Note that questions with the same name are generally different in behavior because they pertain to data segments in different locations.)

1.) *Comparing data points between streams*: The first type of question picks out a single data point and compares its value from data stream 1 and data stream 2. The point can be drawn either from the time series data or the Fourier transformed data; this is specified in each particular question.

Let x_1 be the value of the data point in data stream 1 and x_2 be the value of the data point in stream 2. The algorithm computes the quantity q, given by:

$$q = w_1 \frac{|x_1 - \mu_1|}{\sigma_1} + w_2 \frac{|x_2 - \mu_2|}{\sigma_2} + w_3 \frac{|(x_1 - \mu_1) + s(x_2 - \mu_2)|}{\sigma_3} + w_4 \frac{\sigma_3}{|(x_1 - \mu_1) - s(x_2 - \mu_2)|} \quad (1)$$

The question is answered as 'true' if $q > 1$. Otherwise, the answer is 'false.' The quantities w_1, w_2, w_3, and w_4 are weight factors which are found in the parameters of the chromosome; μ_1 and μ_2 are means from streams 1 and 2 respectively; σ_1 and σ_2 are the standard deviations corresponding to μ_1 and μ_2; and σ_3 is simply the addition in quadrature of σ_1 and $s\sigma_2$, where s is the scale factor for stream 2.

The quantity q is an estimate of how far the data point is above the noise. It should be larger if the data point is significantly above the noise, but it is not obvious exactly how large, and thus the weight factors w_1, w_2, w_3, and w_4 must be optimized by the GA. The scale factor s is intended to account for the different sensitivities of the two detectors, and in order to do this optimally we allow s to be optimized by the GA. The fourth term in equation (1) helps to ensure that the signals we find are in phase between the two data streams. If there is a signal in both data streams, it will interfere destructively between the two data streams.

2.) *Cross-Correlation*: The second type of question looks at cross-correlation coefficients between the streams. Cross-correlation coefficients give an estimate of how similar the two data streams are, and in this sense indicate a signal. The question can operate on either the time series data or the Fourier transformed data, as indicated by the question's individual specifications.

The correlation coefficient is given by

$$CC = \frac{\sum_i \overline{X}_i^{(1)} X_i^{(2)}}{\sqrt{\left[\sum_i \overline{X}_i^{(1)} X_i^{(1)}\right]\left[\sum_i \overline{X}_i^{(2)} X_i^{(2)}\right]}} \tag{2}$$

Here, let $x_i^{(1)}$ be the i^{th} data point in the specified stream 1 data, and similarly for $x_i^{(2)}$ with stream 2. Then $X_i^{(1)} = (x_i^{(1)} - \mu_1)/\sigma_1$ and $X_i^{(2)} = (x_i^{(2)} - \mu_2)/\sigma_2$, where μ_1, σ_1, μ_2, and σ_2 are the corresponding means and standard deviations from streams 1 and 2, respectively. The denominator on the right hand side of equation (2) ensures that $|CC| \leq 1$. The program then checks if $|CC|$ is greater than a threshold. If so, then the question is answered as 'true', otherwise it is answered as 'false'. The threshold is found in the parameters of the chromosome and is constrained between 0 and 1. Since equation (2) involves $X_i^{(1)}$ and $X_i^{(2)}$ rather than $x_i^{(1)}$ and $x_i^{(2)}$, CC will be different for the Fourier transformed data than for the time series data.

3.) *Auto-Correlation*: The third type of question deals with auto-correlation coefficients. These give an estimate of correlations between the data at two different times, and in this sense indicate a signal. The question works in exactly the same way as question 2, except that instead of $X_i^{(1)}$ and $X_i^{(2)}$ we would have $X_i^{(1)}$ and $X_{i+j}^{(1)}$, or $X_i^{(2)}$ and $X_{i+j}^{(2)}$, depending on whether data stream 1 or data stream 2 is specified in the particular question. The integer j represents the lag of the auto-correlation coefficient which must be specified for each auto-correlation question.

4.) *Comparing neighboring data points*: The fourth type of question is implemented in a similar fashion to question 1 in that a quantity q is calculated and the question is answered 'true' or 'false' based on whether $q > 1$. In this case, q is given by

$$q = w_1 \frac{\sigma_1}{|x_1 - y_1|} + w_2 \frac{\sigma_2}{|x_2 - y_2|} + w_3 \frac{|(x_1 - \mu_1) + (y_1 - \mu_1)|}{\sigma_1} + w_4 \frac{|(x_2 - \mu_2) + (y_2 - \mu_2)|}{\sigma_2} \tag{3}$$

Here x_1 is a data point in stream 1, and y_1 is a neighboring data point in the stream j units away from x_1. Similarly, x_2 and y_2 are data points with the same coordinates as x_1 and y_1 respectively, but are drawn from data stream 2. We expect each of the four terms in equation (3) to be large if the neighboring points are correlated, and thus a large q indicates a signal just as large auto-correlation coefficients in question 3 indicate a signal. However, this question, unlike question 3, provides an indication of correlation between two specific data points rather than just looking for a general trend of correlation between neighboring data points. This can be advantageous, for example, in the frequency domain where certain frequencies may be more important than others. The first two terms in equation (3) test anticorrelation similarly to the fourth term in equation (1), whereas the last two terms in equation (3) test correlation similarly to the third term in equation (1).

2.4. Fitness Function

In order to quickly obtain and approximate a numerical value corresponding to the effectiveness of a chromosome in detecting signals in data, we used twenty one-second data clips. A random number of the twenty clips were injected with signals. A chromosome is asked which clips contain signals and which do not, and its fitness is determined by the accuracy of its answers.

In this proof of principle study we chose to use an arbitrary, easy to understand fitness function which has proven useful. A more complicated fitness function, such as using the area under ROC curves, might have improved the evolutionary process, but increases computing time. Specifically, we chose to define our fitness function as:

$$R = M + 3F + e^{2(L-7)} \tag{4}$$

where R is the value of the fitness function (rank), M is the number of missed signals, F is the number of false positives, and L is the number of levels of the tree. The exponential term is only present if L exceeds 7. A fit chromosome has a *small* rank.

If a chromosome misses a large number of signals, it will have a high rank and most likely not pass on its structure to future chromosomes. Chromosomes with a large false alarm rate are penalized even more severely to encourage reliable search algorithms. The third term of the equation is a type of constraint that is referred to as a *penalty function*[11]. Because of computing power restraints, the term was used to restrict the size of chromosomes.

3. Analysis and Results

3.1. Test Data

We have kept the initial run of the GP relatively simple by dividing the data streams into a constant number of data clips of equal length. We used 20 clips that each correspond to 1 second of data. Each of these clips consisted of 16,384 points. For each generation, we randomly sampled 20 seconds of data from a 300 second source of realistic LIGO noise. The realistic detector noise was taken from a previous data run and reflects the characteristic coloring, non-stationarity, and glitchiness of the generic LIGO data, albeit not perfectly. For the development of the production version of the search code, intended to be used for present searches, we will use longer stretches of up to date data.

The simulated signals were then randomly inserted into this sample every generation, but did not overlap into other data clips. We then applied a 60 - 2600 Hz bandpass filter to the data streams. The signals we inserted into the data for this study were Q=8.9, 250Hz sine gaussians of the form:

$$h(t) = h_0 \sin(\omega_\circ t) e^{-\frac{(t-t_0)^2}{\tau^2}} \tag{5}$$

with a central angular frequency of $\omega_\circ = 2\pi f_\circ$, and $Q = \omega_\circ \sigma = 2\pi f_\circ \sigma$ [12].

Sine gaussian waveforms were chosen as a simple and easy to understand example for this proof of principle study. Naturally, search algorithms shall be trained to find different types of signals by simply changing the injected waveform.

We used a dynamic signal scaling system so that if the chromosomes were performing poorly then the scale size of the injected signals was increased, and if they were performing well the scale size was decreased. This method ensured that the injected signals didn't remain too small for good chromosomes to evolve in a practical amount of time, and compelled well-performing chromosomes to become more sensitive.

We used realistic interferometric detector noise as opposed to gaussian noise to evolve the population of our algorithm. We then evaluated the chromosomes on gaussian noise to demonstrate their quality in a simple and transparent test environment. In addition it is customary within the community to perform benchmark tests using gaussian noise.

3.2. Evaluation of the Genetic Algorithm

We ran the program with a population of 40 for 199 generations. Though Haupt[4] recommends a population size ranging from 20-30 for the average GA, we opted for as large a population size as possible given time constraints. This way the GA was better able to sample the vast solution space, thus making it more effective at finding good solutions[13]. It took roughly five days to complete 199 generations on a 64 bit, 1.8 GHz processor with a 1 MB cache and 1 GB DDR PC3200 RAM.

We analyzed a top chromosome from generation 199, which was born during generation 183. In order to demonstrate that evolution occurred, we compared this chromosome and a top chromosome from generation 100. We tested the two chromosomes on gaussian white noise of standard deviation one with injected signals of the form shown in equation (5), and with arbitrarily chosen peak amplitudes of 2, 3.5, and 5 relative to the standard deviation of the noise. The scaling factor between the peak amplitude and the corresponding matched filtering SNR [14] is approximately 9 in this case. Benchmark tests

using ROC curves for each peak amplitude are found in Figure 2. An ROC curve is a measure of how discriminating an algorithm is: the greater the area under the curve, the better the algorithm. A perfect search detects every signal and yields no false positives, resulting in a curve of area 1. An algorithm that chooses randomly is just as likely to have a false positive as it is to correctly detect a signal and would yield a curve of area 0.5.

When detecting 90% of the injected waves, the chromosome from generation 100 had about 2.7×10^7 false alarms per year for peak amplitude 3.5 and about 2.3×10^7 false alarms per year for peak amplitude 5. At the same detection probability, the chromosome from generation 199 had about 1.6×10^5 false alarms per year for peak amplitude 3.5 and about 6.5×10^{-6} false alarms per year for peak amplitude 5. In these two cases the chromosome from generation 199 outperformed the chromosome from generation 100, indicating that chromosomes were improving.

Both chromosomes are mostly made up of cross-correlation questions on untransformed data. The generation 100 chromosome has three such questions and one question of a different type. The generation 199 chromosome has eight such questions out of thirteen total questions. However, the five other questions in the generation 199 chromosome have been found to always yield the same answer. They are effectively 'junk DNA'. When one accounts for the junk DNA it turns out that the tree reduces to one that consists only of OR's and cross-correlation questions on the untransformed data. Thus the generation 199 chromosome simply calculates some cross-correlation coefficients, checks them against a threshold, and if *any* of them exceed the threshold a detection is reported. This is a very simple algorithm that makes intuitive sense, but it is remarkable that the GA was able to evolve a version of this type of algorithm. The generation 199 chromosome was also able to cover most of the range of the data streams with the cross-correlation questions (see below), while optimizing the overlap between cross-correlation windows. Presumably if the evolution had continued the chromosome would have covered the entire range.

The ROC curves in Figure 2 are for signals injected at the same time in the two data streams. We did not intend to evolve algorithms to deal with identifying signals which were not inserted in phase between the two data streams (this is the case for triggered searches, where the time delay between the detectors is well known). In addition, because of the way the signals were inserted, the location of the maximum of any signal was not in the first 418 counts of each one second clip nor in the last 372 counts. Also no cross-correlation question in the generation 199 chromosome examined data located between counts 7988 and 8379 nor between counts 15632 and 16012 - a total blind range of 773 counts. This has been accounted for in the false alarm rates in Figure 2. True positive probabilities are also slightly underestimated since we count missed signals in the insensitive regions of the algorithm.

To produce the ROC curves in Figure 2 only one parameter p was varied. This parameter is the threshold for cross-correlation coefficients on the time series data. All other parameters were kept at their optimized values.

4. Conclusion

The complex nature of the LIGO data makes EC an interesting candidate for development of a gravitational wave search algorithm. The power of this method was displayed in its ability to develop useful search algorithms for coincident short burst signals in a short time.

The questions of the best chromosome of the run (see Figure 1) are very simple. The evolutionary program's ability to create effective solutions from elementary building blocks indicates its potential to create alternative gravitational wave search algorithms for LIGO data as well as its possible use for veto studies.

Future runs of the evolutionary program will be done on a computing cluster, which will provide an increased source of computing power. We will utilize this power by expanding the size of the solution space through an increase in the maximum number of levels allowed for a tree before it is penalized by the fitness function. There are also several other changes such as allowing more parameters to be optimized, increasing the population size, injecting multiple signal types, and defining a more objective

fitness function that will be made in order to take advantage of the increased computing power. In addition we intend to add to the list of possible questions. For example we will allow for questions about matched filtering statistics.

Most importantly, we plan to run the evolutionary program until chromosome improvement stops. We also plan to enable the chromosomes to examine n data streams. In the case of many detectors and complex events, an evolutionary program might very well be the best way to come up with an n-detector coherent analysis. We believe that future runs of the program will continue to yield interesting results with significantly better sensitivity and lower false alarm rates.

5. Acknowledgements

The authors gratefully acknowledge the support of the United States National Science Foundation and Columbia University in the City of New York, which made this research possible.

References

[1] Hughes, S. A., Márka, S., Bender, P. L. and Hogan, C. J. 2002, in *The Future of Particle Physics: 2001 Snowmass Meeting*, ed. N. Graf, (astro-ph/0110349)

[2] Abbott, B. et al. 'Detector Description and Performance for the First Coincidence Observations between LIGO and GEO.' *Nucl. Instrum. Methods A*. Vol. 517. 2004.

[3] Eiben, A. E. and J. E. Smith. *Introduction to Evolutionary Computing*. Berlin: Springer, 2003.

[4] Haupt, Randy L. and Sue Ellen Haupt. *Practical Genetic Algorithms*. 2nd ed. Hoboken, NJ: Wiley Interscience, 2004.

[5] Mitchell, Melanie. *An Introduction to Genetic Algorithms*. Cambridge, MA: A Bradford Book, 1996.

[6] Lohn, J. D., Linden, D. S., Hornby, G. S., et al. 'Evolutionary Design of an X-Band Antenna for NASA's Space Technology 5 Mission.' *Proc. 2004 IEEE Antenna and Propagation Society International Symposium and USNC/URSI National Radio Science Meeting*. Vol. 3. 2004. 2313-2316

[7] Dasgupta, D. and Z. Michalewicz, ed. *Evolutionary Algorithms in Engineering Applications*. Berlin: Springer, 1997.

[8] Davis, Lawrence, ed. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991

[9] Reeves, Colin R. and Jonathan E. Rowe. *Genetic Algorithms - Principles And Perspectives: A Guide to GA Theory*. Boston: Kluwer Academic Publishers, 2003.

[10] We have left open the possibility of a 'not' operator. This would be useful for veto studies.

[11] Colely, David. *Introduction to Genetic Algorithms for Scientists and Engineers*. Singapore: World Scientific Publishing Co., 1999.

[12] Abbott, B. et al. 'Search for gravitational waves associated with the gamma ray burst GRB030329 using the LIGO detectors.' *Physical Review D* 72.4 (2005): 042002.

[13] Cantú-Paz, Erick. *Efficient and Accurate Parallel Genetic Algorithms*. Boston: Kluwer Academic Publishers, 2000.

[14] Flanagan, E.E., and Hughes, S.A. 'Measuring gravitational waves from binary black hole coalescences. I. Signal to noise for inspiral, merger, and ringdown.' Phys. Rev. D. Vol. 57. 1998. 4535-4565

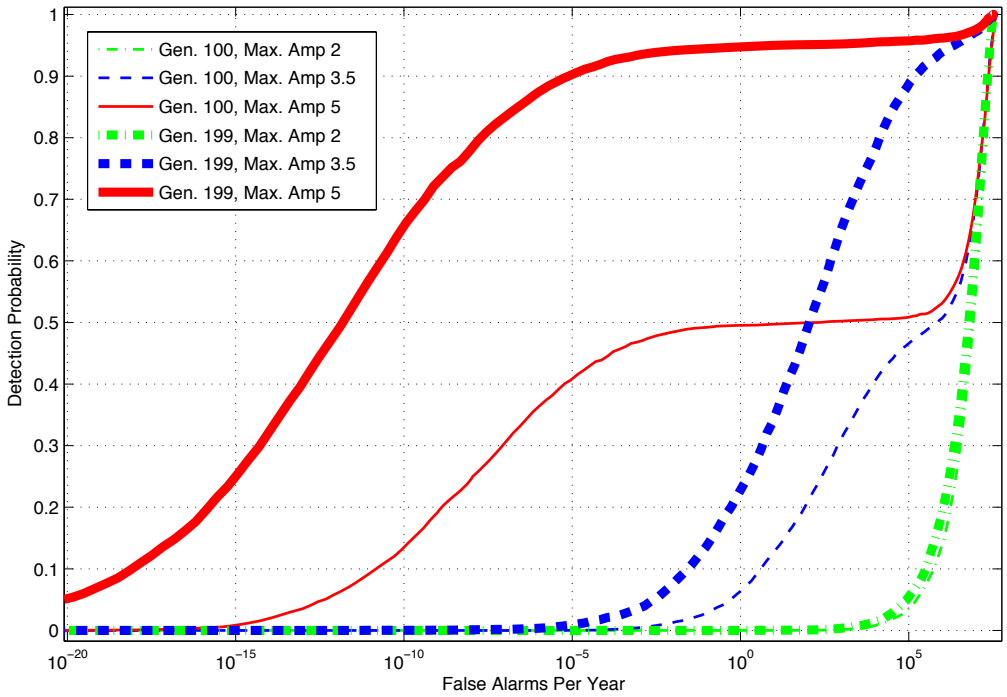

Figure 2. Above are the Receiver Operating Characteristic (ROC) curves for representative chromosomes in generations 100 and 199 on signals with peak amplitudes of 2, 3.5, and 5 relative to the standard deviation of the noise. Observing the difference between the thick and thin lines, evolution has clearly taken place and significant improvement has occurred.