# Performance of Diploid Dominance with Genetically Synthesized Signal Processing Networks

**F. Greene**
**6920 Roosevelt NE #126**
**University of Washington**

**Abstract**

A methodology is described for synthesizing signal processing networks, which are used to solve a low-cost medical signal processing problem. The approach makes use of genetic algorithms and a new approach to diploid/dominance, which is tested using clinical patient data. Two complementary reasons for an observed diploid efficiency increase are proposed: 1) Improvements due to the retention of relatively low-fitness, recessive building blocks and 2) Improvements due to the increased proportion and fast evaluation of non-viable, recessive genotypes.

## 1. Introduction

### *1.1. Problem Definition and Summary*

Networks that process information have been synthesized using *Genetic Programming*. These have been the subject of a number of authors, beginning with Cramer (1985) and extensively developed by Koza (1993) and others. This paper uses a similar approach to solve a low-cost medical instrumentation problem, and at the same time extends previous work by Greene (1994) that indicated potential GA performance improvement using stationary diploid dominance.

The approach used here expands a real-valued genotype, depth first, into a GP-like network structure using a pre-defined repertoire of node functions, known as the *function set*. This set of inter-connectable nodes consists of both standard arithmetic and Boolean, as well as more specialized signal processing nodes. The result is a network system capable of processing noisy, spectrum analyzed audio data in real-time.

The use of complete diploid dominance was previously shown by Greene (1996) to provide a discernible efficiency improvement with a deceptive, multimodal and moreover *stationary* fitness function. Theory, based on population genetics, was also used to show how such performance improvement might occur with GA problems that have reasonable deceptivity. That work is extended here in a multi-trial comparison with a network synthesis problem.

For clinical data analysis, the portion of a patient's ECG called the *R-Wave* can be useful as a fiducial point. Providing an R-Wave substitute, based exclusively on the Doppler blood flow signal, is the medical instrumentation design problem adopted here. A GP-like network of processing elements will be synthesized that produces an output in synchrony with the recorded R-Wave. This particular problem has plentiful data available for training and testing, since every heartbeat and associated ECG of a recorded Doppler signal provides a test case. Also, a Doppler based fiducial point is convenient in fetal heart monitoring. Primarily however, a reliable R-Wave substitute requires robustness over a wide variety of signal conditions, including signal dropouts and other artifact, and demonstrates a basic capability needed for more advanced disease quantification, flow/no flow detection, etc.

### *1.2. Doppler Flow Signal Processing*

Schematic examples of Doppler flow data are shown in figure 1. The three graphs depict one heart beat of data each, and plot the upper and lower 9 dB sidebands vs. time. The graphs show laminar, or *normal* flow, turbulent (broad band) flow, and turbulent flow with an intervening signal dropout artifact. As suggested by the annotation in this figure, one of the problems addressed in the implementation here will be the detection of both signal dropout (to detect unusable data), and in addition the discrimination of broad band vs. laminar flow at a given point in time. The latter amounts to a decision made on a per spectrum basis, and is used to improve calculated accuracy of the upper 9 dB sideband.

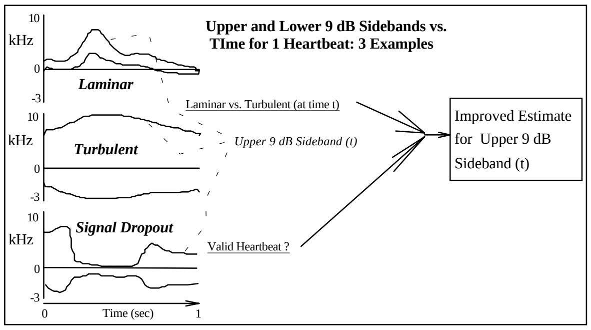


Figure 1.1. Pulsed Doppler signal flow states. Three example heartbeats showing different flow states.

A fairly typical Doppler spectrum, computed from one 40 msec window within a single heartbeat, is shown in figure 1.2. The nature of the Doppler instrumentation permits quadrature detection, which in effect provides positive and negative frequency information. These correspond respectively, to flow components (within a *sample volume* in space) that are toward and away from the (hand-held) Doppler probe. Subsequent to calculating the FFT periodogram, the upper and lower 9 dB sideband frequencies (as measured from the spectral peak or *mode*) can be defined and measured as illustrated. Also indicated in this figure is the potential for multiple crossings in the 9 dB upper sideband calculation. If the spectrum is determined to be broad band, as mentioned above, the upper sideband calculation is modified so that the first 9 dB crossing coming *down* from the maximum possible positive frequency is used. Otherwise, for laminar flow the first crossing coming *up* from the spectral mode is used for the upper sideband.

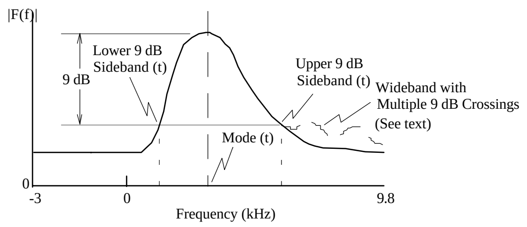
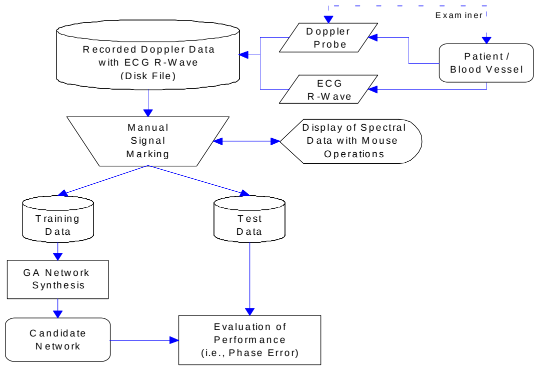


Figure 1.2 A typical Doppler signal periodogram at time t, including the definition of spectral mode and lower and upper 9 dB sideband.

## ***2. Methods***

Networks are created that consist of interconnected *nodes*, each of which can be one of a number of different functionalities, or *types*. The set of node "types" that are available for instantiation at a given location in the network is referred to as the "function set" by Koza (1993). A portion of this function set consists of nodes that implement frequency-domain based (FFT) functions, some of which were indicated to be useful in Doppler signal processing by Greene (1988). These include spectral energy, mode and 9 dB upper side-band calculation. The 9 dB calculation has an adaptive characteristic that makes it more usable in adverse S/N situations, as explained below. A non problem-specific portion of the function set includes a *targeting* node that can form both feedback as well as feed forward signal path connections. An ADF (Automatic Function Definition) operator, as developed by Koza (1993), was also included. A small, additive component in the fitness function is subsequently derived from the output of this ADF, for the purpose of training it to help in detecting and hopefully aiding in the rejection of flow signal noise and artifact.


Examiner
Doppler Probe
Recorded Doppler Data with ECG R-Wave (Disk File)
Patient / Blood Vessel
ECG R-Wave
Manual Signal Marking
Display of Spectral Data with Mouse Operations
Training Data
Test Data
GA Network Synthesis
Candidate Network
Evaluation of Performance (i.e., Phase Error)
Figure 2.1. Problem definition and approach.

A GA will be used to train a network of processing elements, that is similar to an electrical circuit, to produce an output that mimics the ECG R-Wave. The essential aspects of the approach are shown in figure 2.1. Doppler flow signals, recorded simultaneously with the R-Wave as described above, will provide the training data for this purpose. These recordings will then be manually marked to indicate signal irregularities, missing R-Wave's, etc. These data will then form the training set for network synthesis, together with a network performance metric, which is primarily the phase error between the network and R-Wave outputs.

### *2.1. GA and Function Set Description*

The Genitor GA, developed by Whitley (1989), was adapted to C++ with floating point chromosomes. C++ objects were defined to implement the network nodes. Genitor utilizes *decreasing* fitness to indicate *increasing* optimality. Consistent with this, zero

Genitor fitness corresponds to the global optimum, and this is appropriate for minimizing an *error* criterion such as used here. C++ classes were added to Genitor to define network nodes and the diploid chromosome.

All chromosome loci are real-valued and restricted to the range [-1..1]. Instantiation of a particular node's type is specified by a single corresponding allele value on the chromosome. In addition, 0 to 4 *node operands* may immediately follow the node type and be used to parameterize the node's behavior. The number of such "node operands" is pre-determined for each node type. Crossover is restricted so that it only occurs *between* nodes and never within a node and its operands. Real-valued mutation, similar to that used by Michalewicz (1992, pg. 88), can occur at *any* locus.

Also defined for every node are *constructor* and *eval()* functions, or *methods*. The constructor controls memory allocation and variable initialization. The "eval()" method is called when it is desired to have the node "evaluate" the output(s) of its child node(s) or other input data. Each node provides a real scalar output to its parent, and expects 0 or more children depending on its predefined node *arity*. Certain problem-specific *frequency domain* nodes can also access a global register that contains the most recently calculated Doppler (FFT) spectrum.

Every Doppler FFT provides 64 points of magnitude data from 40 msec of successive, but non-overlapping, digitized Doppler time segments. The global FFT register is updated with the spectrum of the next time window, just prior to network evaluation. The "frequency domain" nodes have no children, and are what are known as *terminal* nodes since they will end the current network branch when instantiated. Other "terminal" nodes include a NULL node and a TAR (targeter) node. The null node always outputs a 0, unless its parent is an AND node in which case it outputs floating point infinity.

The "TAR" node makes it possible to have arbitrary input/output connections between any two nodes in the network. A single node operand for this node specifies an overall network target region, relative to the depth of the TAR node itself[1]. The sign of this operand determines whether the connection will be feed back or feed forward. In a second pass after network instantiation, the closest network node to each TAR node's target "region" is assigned as the TAR node's signal input.

The remainder of the function set consists of relatively general purpose nodes. These include fuzzy Boolean (AND, OR, INVERT) nodes plus the four arithmetic operators: +-*/. Also included are a differentiator node, a single operand variable delay node and a single argument ADF (designated ADF1). The ADF has a separate chromosome, so that crossover exchanges building blocks that don't compete with main function building blocks.

*2.2. Specialized Function/Terminal Set Nodes*
Although the "frequency domain" nodes use the current spectrum for their input data, they still produce a real scalar output and so can be the child of any non-zero arity node. These nodes are as follows (including their reference mnemonics):

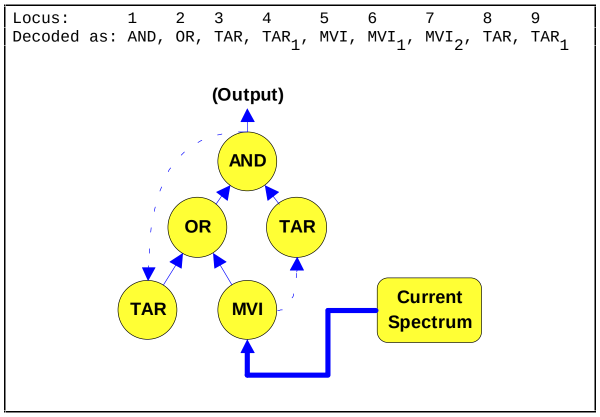
- MVI returns the spectral *mode* (peak frequency) between two operand specified band limits.
- ENR returns the spectral energy. There are 4 operands that specify positive and negative frequency band limits.
- 9DB returns the upper 9 dB sideband. The 9 dB level is calculated from the spectral mode. For narrow band data the 9 dB frequency is the 1st 9 dB crossing going *up* from the mode. For wide band data, the *last* 9 dB crossing is used. The narrow/wide band decision is made by counting the number of bins above a histogram defined threshold. This count is then compared with an operand defined value.

In addition to the above, an arity-1 *Onc-shot* (OS) node produces a pulse when its child produces an input greater than an operand defined threshold. The pulse width is the same as the time between spectra, or 40 msec.

*2.3. Network Expansion and Fitness Evaluation*
The real-valued genotype is read in ascending index order to create the network recursively, in depth-first order. The C++, GP-like approach for doing this was

[1]Here, depth refers to the distance from the network root node to the root node's most distant ancestor.

described by Keith (1993).



```
Locus:       1    2    3    4     5    6     7     8    9
Decoded as: AND, OR, TAR, TAR1, MVI, MVI1, MVI2, TAR, TAR1
```

Figure 2.2. A simple chromosome, whose *real-valued loci* decode into the indicated nodes and node operands. Subscripts (e.g., "$TAR_1$, $MVI_1$, $MVI_2$") denote node *operands*, which are themselves real-valued chromosome loci that immediately follow each node type locus (see text).

Network fitness is evaluated by first determining if there is at least an OS and MVI node in the network. If this *viability* criterion is not met, fitness evaluation stops and the network is assigned minimal fitness. Otherwise, the network is evaluated using the spectral training data. This is done by firing the root node's "eval()" method once for each signal spectrum. Network outputs are compared to the recorded R-Wave to obtain the average phase error. To this score is added a penalty of 1000 for each missing trigger. It is also multiplied by the square of the number of triggers per each R-Wave.

In order to encourage resistance to signal dropouts and noise, a second component to the performance evaluation is introduced. This is a small additive term that penalizes fitness according to the ADF output.

## *2.4. Implementation of Diploid Dominance*

*Complete* diploid dominance is implemented in an identical fashion to that described in Greene (1996). Dominance is termed "complete" because the heterozygote fitness has the same fitness as the homozygous dominant, as discussed by Felsenstein (1995). The fitness of each diploid individual is defined to be the maximum fitness of its two, independently evaluated homologues.

## **3. Test Data**

### *3.1. Clinical Doppler Signal R-Wave Recognition*

In the GA runs used here, two 12.5 sec records from a normal and diseased carotid artery were combined to form a single 25 second record. The result of this edited combination are the data shown in figure 3.1, below. This data set exhibits signal variability and the relatively broad band characteristics associated with moderate arterial disease.

Superimposed over the Doppler signal data of figure 3.1 are the manually generated "marks" used to indicate where triggers *must* occur. A fitness penalty of 1000 is assessed for every missing trigger in a marked heartbeat. Two other "marks" indicate: 1) Where a trigger may *optionally* occur (e.g., if the ECG produced no R-Wave trigger) and 2) "Train to reject", indicating signal dropout, excessive noise, etc. This latter mark is used solely as a reference signal that the ADF is trained to mimic[2]. The vertical lines show the ECG R-Wave locations. If a network produces exactly one output per heartbeat, the network fitness is the average phase error between network triggers and the nearest R-Wave. A good network fitness corresponds to a phase error of approximately 1.0 or less.

In an order to minimize run time, the initial population for each of the 15 runs was *back code* seeded with a single individual whose genotype produces the network shown in Appendix figure A.1.1. "Back coding", as used here, designates an encoding algorithm that generates the required genotype for a desired phenotype. This was implemented with C code that is controlled by a simple network definition text file. An intentional characteristic of the back coded network of figure A.1.1 is the presence of a *single* ADF call.

### *3.2. Effect of Diploidy*

The effect of diploidy on GA efficiency was investigated by comparing multiple haploid and diploid GA runs. These runs all used identical selection, mutation and crossover parameters, and where possible, identical initial populations. Specifically, the diploid homologues of each individual in the initial populations were initialized with *identical* genotypes. In addition, each

[2]The MSE (Mean Squared Error) between the ADF output and the 1st derivative of the reject marks is scaled and added to the final fitness score.

haploid/diploid trial used the same random number seed to define the initial population. 5 trials were done with haploid N=100 and diploid N=100 each. Also, 5 trials were done with haploid N=200, in an attempt to bracket the diploid comparison with a haploid population having more initial diversity. (One half of the initial N=200 haploid population was also genotypically identical with the haploid and diploid, N=100.)

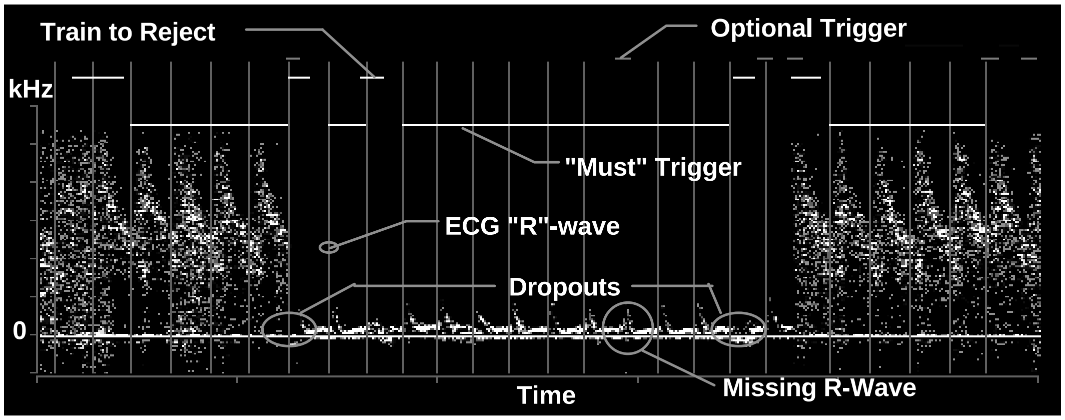


Figure 3.1 Training data set (see text). The vertical lines are ECG R-Wave events, and the horizontal lines are manually generated signal condition marks, as described in the methods section.

## 4. Results

### *4.1. Analysis of Network Operation*

A potential virtue of the high level of node functionality used here, is that network operation can be analyzed and given an intuitive rationale. That turned out to be the case for the clinical R-Wave problem solution.

An overall sense of how one synthesized network operates is shown in figure 4.1. (The actual network is shown in figure A.1.2, using the function set mnemonics described in the methods section.) This example had an average phase error of 0.62. The ADF(1) gets called twice in the main network: Once with a 9 dB input and once with a MVI (spectral mode) input. If the 9 dB value at the input to its ADF1 drops, due to a signal abnormality, its sibling MVI leg can prevent a collapse to zero.

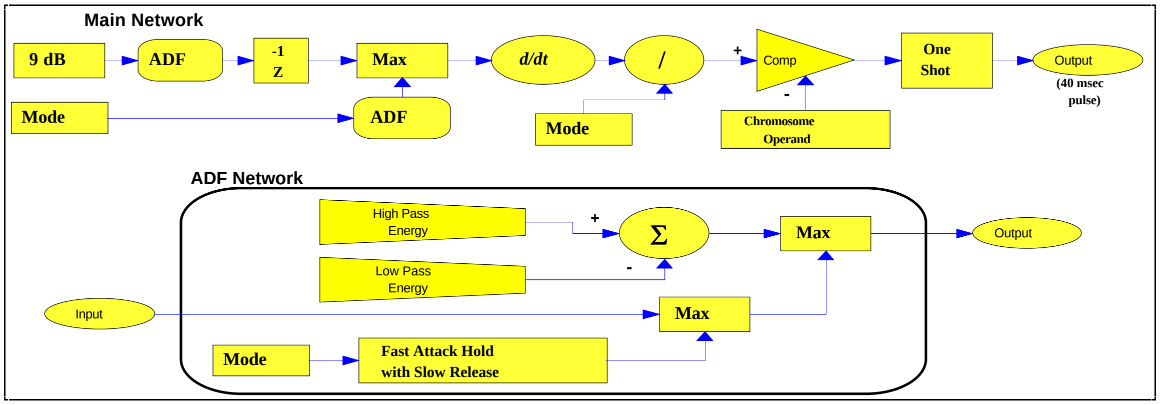


Figure 4.1. Simplified, complete schematic diagram of both the main and ADF networks for one solution. The "Mode", "9 dB" and "Energy" nodes take their input from the current spectrum.

In principle, at least, the presence of two calls in the network to the ADF1 function illustrates what Koza calls *scalability*, or the re-use of evolved functionality. The maximum of the two ADF outputs are differentiated and divided by the spectral mode. This scaling provides some compensation for wide variations in the frequency excursions associated with various disease states, as represented in the training data of figure 3.1.

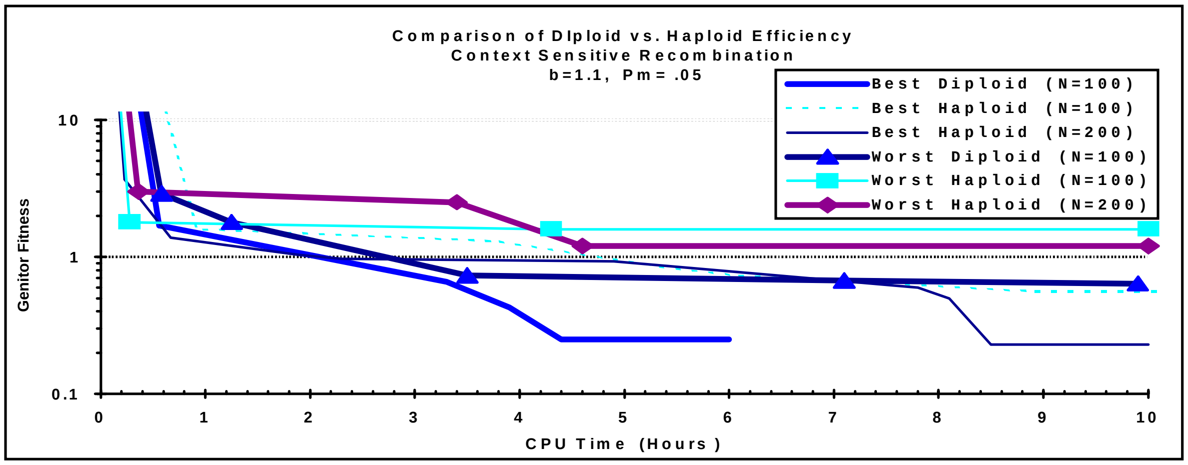


Figure 4.2. Comparison and summary of best and worst of five diploid and haploid convergence times. The data are interpolated for clarity.

*4.2. Multiple Trial Diploid vs. Haploid Comparisons*
Comparisons for five diploid, five haploid N=100 and five haploid N=200 runs are shown in figure 4.2, using the spectral training data of figure 3.1. Back coding was used to seed one individual into each initial population. Genitor fitness scores below 1.0 are considered to have low network output phase error for the purpose of identifying success.

In summary, all diploid runs achieved a fitness of 0.73 or less, in under 4 hours of run time. By contrast, 3 of the haploid N=100 runs and 2 of the haploid N=200 runs failed to achieve a fitness of 1.1 in over ten hours (at which time the runs were terminated). Of the haploid, N=200 runs, only one achieved a fitness better than the *worst* diploid (this haploid run is the one shown as the "best haploid" for this group). In this respect, diploidy appears to provide some protection against relatively long GA runs. Better diploid than haploid efficiency is also seen when the *worst* diploid and haploid N=200 (and haploid N=100) runs are compared, in figure 4.2 above.

There were (haploid) runs at population sizes of both N=100 and N=200 that did *not* reach a reasonable solution in the pre-allotted 10 hours. Therefore, average fitness comparisons are not possible. By defining a good fitness result to be 0.75 or better, which corresponds to an average network phase error of approximately 30 msec, the *range* of run times can be compared. This result is summarized in table 4.1. As might be expected, the haploid N=200 runs showed generally better performance than the haploid N=100 runs.

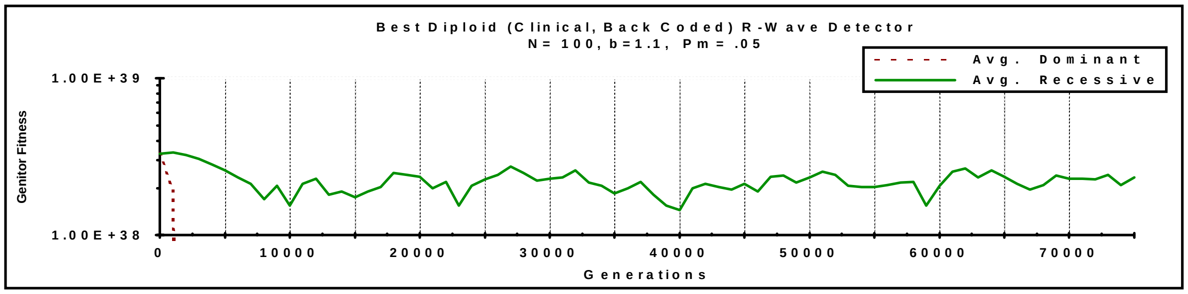


Figure 4.3. Average recessive fitness for one diploid, network synthesis GA run. The presence of low fitness genotypes are consistently maintained throughout.

Average recessive fitness remains within an order of magnitude of the minimum possible Genitor fitness (plus infinity) throughout the entire run. The average fitness evaluation rate for the above haploid runs was calculated to be 5.6 seconds, compared to 4.1 seconds for diploid. This gives a ratio of average diploid to haploid function evaluation times of 2· (4.1/5.6)= 1.46. This ratio is greater than 1.0, which is to be expected since diploidy will shield non-viable, fast to execute genotypes from selection. The capacity for diploidy to maintain continual proportions of low-fitness genotypes is evident in figure 4.3, above.

Table 4.1. Best and worst evaluation times, in hours, for 5 training runs of the clinical R-Wave problem. Run time is defined by the point in run time when a Genitor fitness of 0.75, or better, is reached. Since best run times are available for all haploid, as well as diploid tests, these are highlighted.

| | Diploid | Haploid, N=100 | Haploid, N=200 |
|---|---|---|---|
| **Best** | **3.3** | **6.0** | **7.8** |
| Worst | 3.3 | > 10 | > 10 |

## 5. Conclusions

Diploid dominance is implemented using a stationary fitness criterion, and utilized with a medical signal processing, network synthesis problem. The experimental results indicate a slight but consistent increase in GA efficiency when comparing complete diploid dominance with haploidy. Average diploid efficiency improvement with previously published multimodal trials was approximately 2:1, and is comparable to the results seen here. A portion of the observed efficiency improvement was seen to occur due to low fitness, fast to evaluate genotypes.

This approach to diploidy is demonstrated to *not* be problem specific, and appears to be helpful with a problem of moderate complexity. In particular, a relatively large function set was utilized, including approximately 20 function types. This function set includes both fairly general purpose and problem specific function definitions. One of these functions (the "TAR" node) is designed to create both feed back as well as feed forward network connections, and was retained by the GA in final solutions, one of which is shown.

## Bibliography

Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs. In Grefenstette (Ed.), First International Conference on Genetic Algorithms, (pp. 183-187). Pittsburgh: Erlbaum.

Felsenstein, J. (1995). Theoretical Evolutionary Genetics. Seattle: ASUW Publishing.

Greene, F., Beach, K., Phillips, Primozich, J., Strandness, D. (1988). Pattern recognition of carotid disease using pulsed Doppler ultrasound. In IEEE EMB Society 10th Annual Conference., 10 (pp. 229).

Greene, F. (1994). A method for utilizing diploid/dominance in genetic search. In World Conference on Computational Intelligence, ICEC-1 (pp. 439-444). Orlando, FL: IEEE.

Greene, F. (1996). A new approach to diploid/dominance and its effects on stationary genetic search. In D. Fogel (Ed.), Proceedings of the Fifth Annual Conference on Evolutionary Programming, . San Diego: MIT Press.

Keith, M. J., Martin, M.C. (1993). Genetic Programing in C++ No. Allen Bradley Corporation, Heighland Heights, Ohio, 44143, Keithm@odin.icd.ab.com, (216) 646-3464.

Koza, J. (1993). Genetic Programming (3rd Printing ed.). Cambridge: MIT Press.

Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. Berlin: Springer-Verlag.

Whitley, D. (1989). The Genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In Proceedings of the Third International Conference on Genetic Algorithms, (pp. 116-123). George Mason University: Morgan Kaufmann.

## Appendix

*Synthesized Networks*

```
OS( 5.0, 1, 1.0)0<-0<-OR(2)1<-0<-FD(0.67,0.10,0)2<-0<-AND(2)3<-0<-Dly(0,1)4<-0<-Inv5<-0<-WR(0)6<-0<-9DB7<-0<-MVI(
                                                                                                                        1<-W/N9
                                                               1<-ADF111<-0<-RD(0)13<-
                              1<-Null14<-
```

Figure A.1.1. Network used to backcode initial population for solution #1.

```
OS( 0.5, 1, 1.0)0<-0<-DIV1<-0<-DFR(0.99,6)2<-0<-OR(2)3<-0<-Dly(0,1)4<-0<-ADF15<-0<-WR(0)14<-0<-9DB15<-0<-MVI(0,48
                                                                                                    1<-W/N17<-0
                                                                                                              1
                                                        1<-ADF120<-0<-MVI(0,46)21<-
                            1<-TAR(16)22<-
ADF1 subnet/function:
NAND(2)6<-0<-SUB7<-0<-ENR(6,19,62,55)8<-
                   1<-ENR(37,47,61,51)9<-
          1<-NOR(2)10<-0<-ARG111<-
                       1<-FD(0.84,0.06,0)12<-0<-MVI(0,47)13<-
```

Figure A.1.2. Clinical Doppler R-wave problem solution #2 including single argument ADF sub network.