J. Lohn, J. Crawford, A. Globus, G. Hornby, W. Kraus, G. Larchev, A.Pryor, D. Srivastava, "Evolvable Systems for Space Applications," International Conference on Space Mission Challenges for Information Technology (SMC-IT 2003), Pasadena, July 13-16, 2003, to appear.

Evolvable Systems for Space Applications

Jason Lohn[1], James Crawford[1], Al Globus[2], Gregory Hornby[3], William Kraus[3],
Gregory Larchev[3], Anna Pryor[1], Deepak Srivastava[2]
[1]Computational Sciences Division, NASA Ames Research Center, Mountain View, CA 94035
[2]Computer Sciences Corp., NASA Ames Research Center, Mountain View, CA 94035
[3]QSS Group, Inc., NASA Ames Research Center, Mountain View, CA 94035
Email: Jason.D.Lohn@nasa.gov

Abstract

This article surveys the research of the Evolvable Systems Group at NASA Ames Research Center. Over the past few years, our group has developed the ability to use evolutionary algorithms in a variety of NASA applications ranging from spacecraft antenna design, fault tolerance for programmable logic chips, atomic force field parameter fitting, analog circuit design, and earth observing satellite scheduling. In some of these applications, evolutionary algorithms match or improve on human performance.

1 Spacecraft Antenna Design

In this section we summarize a proof-of-concept study [12] that investigated the optmization of the deployed antenna on the Mars Odyssey spacecraft.

Automated antenna synthesis via evolutionary design has recently garnered much attention in the research literature [13]. Evolutionary algorithms show promise because, among search algorithms, they are able to effectively search large, unknown design spaces.

NASA's Mars Odyssey spacecraft is currentlyin Martian orbit. Onboard the spacecraft is a quadrifilar helical antenna that provides telecommunications in the UHF band with landed assets, such as robotic rovers. This antenna can be seen in Fig. 1. Each helix is driven by the same signal which is phase-delayed in 90° increments. A small ground plane is provided at the base. It is designed to operate in the frequency band of 400-438 MHz.

Based on encouraging previous results in automated antenna design using evolutionary search, we wanted to see whether such techniques could improve upon Mars Odyssey antenna design. Specifically, a coevolutionary genetic algorithm is applied to optimize the gain and size of the quadrifilar helical antenna.

Figure 1: Photograph of the quadrifilar helical UHF antenna deployed on the Mars Odyssey spacecraft.

The optimization was performed *in-situ* – in the presence of a neighboring spacecraft structure [9]. On the spacecraft, a large aluminum fuel tank is adjacent to the antenna. Since this fuel tank can dramatically affect the antenna's performance, we leave it to the evolutionary process to see if it can exploit the fuel tank's properties advantageously.

Optimizing in the presence of surrounding structures would be quite difficult for human antenna designers, and thus the actual antenna was designed for free space (with a small ground plane). In fact, when flying on the spacecraft, surrounding structures that are moveable (e.g., solar panels) may be moved during the mission in order to improve the antenna's performance.

1.1 Experiments and Results

Experiments were set up as follows. The Numerical Electromagnetics Code program was used to evaluate all antenna designs. We used a parallel master/slave generational genetic algorithm with a population size of 6000. One point crossover across byte boundaries was used at a rate of 80%. Mutation was uniform across bytes at a rate of 1%. Runs were executed on 32-node and 64-node Beowulf computing clusters.

The wire geometry encoded by each individual chromosome was first translated into a NEC input deck, which was subsequently sent to the NEC simulator.

The segment size for all elements was fixed at 0.1λ, where λ was the wavelength corresponding to 235 MHz.

A coarse model of the neighboring fuel tank was used in the simulations. Its size and position was calculated based on engineering drawings of the spacecraft. To compare our results to the spacecraft antenna, we modeled that antenna with the best data we had at the time of this writing.

A coevolutionary genetic algorithm was applied to the quadrifilar helical antenna optimization. Two populations are used: one consisting of antenna designs, and one consisting of target vectors. The fundamental idea is that the target vectors encapsulate level-of-difficulty. Then, under the control of the genetic algorithm, the target vectors evolve from easy to difficult based on the level of proficiency of the antenna population.

Each target vector consists of a set of objectives that must be met in order for a target vector to be "solved." A target vector consisting of two values: the average gain (in dB), VSWR, and antenna volume. A target vector was considered to be solved by a given antenna if the antenna exceeds the performance thresholds of all target.

Values for target gain ranged between -50 dB (easy) and 8 dB (difficult). Target VSWR values ranged between 100 (easy) and 20 (difficult). Target antenna volumes ranged from 100,000 cm^3 (easy) to 100 cm^3 (difficult). Target vectors are represented as a list of floating point values that are mutated individually by randomly adding or subtracting a small amount (5% of the largest legal value). Single point crossover was used, and crossover points were chosen between the values.

Antennas are rewarded for solving difficult target vectors. The most difficult target vector is defined to be the target vector that only one antenna can solve. Such a target vector garners the highest fitness score. Target vectors that are unsolvable, or are very easy to solve by the current antenna population, are given low fitness scores.

Fitness was expressed as a cost function to be minimized. The calculation was as follows:

$$F = -G_L + \sum (C * V_i) \qquad C = \begin{cases} 0.1 & \text{if } V_i \leq 3 \\ 1 & \text{if } V_i > 3 \end{cases}$$

where: G_L = lowest gain of all frequencies measured at $\theta = 0^\circ$ and $\phi = 0^\circ$, V_i = VSWR at the ith frequency. Lacking from this calculation was a term involving sidelobe/backlobe attenuation. We chose not include such a term because we reasoned that as the mainlobe gain increased, the sidelobes/backlobes would decrease in size.

A set of five runs were executed using the algorithm described above. Only one of the runs found an antenna design that exceed that benchmark antenna. Fig. 2 shows the gain plots for both the evolved and actual Mars UHF antennas. Fig. 3 show the antennas, structures, and radiation patterns of actual Mars Odyssey UHF and evolved antenna. The evolved antenna measures 6cm × 6cm × 16cm which approximately four times as small volumewise as the benchmark (roughly 10cm × 10cm × 25cm). At 400 MHz, the average gain of the evolved antenna was 3.77 dB and 1.95 for the benchmark antenna. At 438 MHz, the average gain of the evolved antenna was 2.82 dB and 1.90 for the benchmark antenna. This represent a 93% improvement at 400 MHz and a 48% improvement at 438 MHz in the average gain.

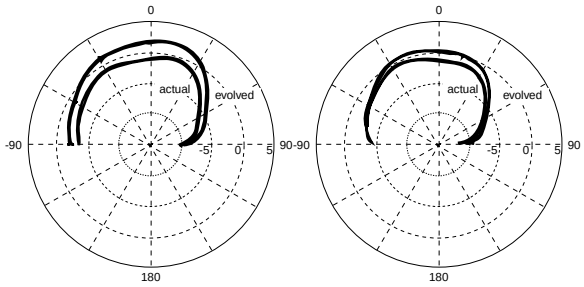

Figure 2: Gain plots for 400 MHZ (left) and 438 MHz (right). In each case, the evolved antenna maintains a higher gain than the actual Mars Odyssey antenna. Plots take into account circular polarization.

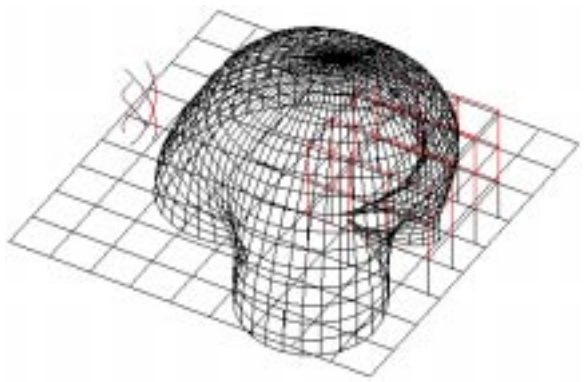

Figure 3: Radiation pattern of the evolved antenna. The antenna can be seen in the upper left and the fuel tank in the lower right.

1.2 Discussion

An improved version of the quadrifilar antenna currently flying on Mars Odyssey was presented. The evolutionary algorithm allowed the antenna to be designed in the presence of the surrounding structure, whereas the human-designed antenna was designed for free-space. Results showed a 93% improvement at 400 MHz and a 48% improvement at 438 MHz in the average gain. The evolved antenna was also one-fourth the size of the actual antenna on the spacecraft, which is important because of the scarcity of area on spacecraft.

For human antenna designers, designing an antenna to be synergistic with its surrounding structures is typically a daunting task. The results from the quadrifilar helical antenna provide encouraging evidence that evolution can exploit those structures to give increased antenna performance.

2 Atomic Force Field Parameters

In this section, we summarize work on automatically finding molecular force field parameters [3].

Accurate molecular dynamics simulation of reactive systems containing many atomic species is important for the conceptualization, design and testing of novel nanoscale materials, molecular electronic devices, nano-integrated systems and applications, and a broad range of physical and chemical phenomenon in other areas as well. The physical and chemical characterization of carbon nanotubes and fullerenes, design and operations of molecular gears, hinges, three-way junctions, and bearings have also utilized simulations using reactive dynamics of 2- or 3-atomic species containing systems [2]. However, as the system and device sizes continue to shrink and composition becomes more multi-species, there is an urgent need for developing good quality reactive atomic force field functions that are not currently available.

The primary impediment to determining the potentials is simulation speed. Simulation at the quantum-mechanical level is prohibitively slow for more than a few hundred atoms. However, millions of atoms can be simulated using classical potentials, albeit with less accuracy, and this is the approach taken here. Unfortunately, reactive multi-species potentials are only available for a few atomic species. Furthermore, developing reactive multi-species potentials is difficult, time consuming, tedious, failure prone and, thus, rarely attempted.

There are two parts to developing atomic force field functions. First, finding an analytic functional form that reflects the physical and chemical nature of the atomic species under consideration, and second, fitting parameters in a complex multi-dimensional parameter space based on the data available from the experiments or more accurate quantum mechanical calculations. In an ideal case, the cycle of choosing a functional form and parameterization of the force field function should be iterated until a reasonable convergence on the choice of inter-atomic potentials is achieved. Doing this for multi-component systems is extremely tedious because the parameter space that needs to be investigated is large and may be coupled in a complex way. The tedium has deterred regular successful attempts in developing

2.1 Evolutionary Algorithm

JavaGenes [6] is a steady state tournament selection genetic algorithm. The tournament size is usually two. In tournament selection each parent is chosen by randomly selecting two individuals from the population and choosing the fittest to be the parent. After crossover or mutation produces a child, individuals to replace are chosen by an anti-tournament of size two. An anti-tournament chooses the least fit individual.

We represent force field parameters as a ragged two-dimensional array of double precision floating point numbers. The first dimension represents the two- or three-body terms of the potential function, and the ragged second dimension holds the varying number of parameters depending on the number of bodies. Each parameter is assigned a set of limits within which it is allowed to evolve. The limiting values of the parameters are chosen from the physical interpretation of the contribution of the parameter to the force field function and are randomized among jobs.

Evolution is guided by a fitness function. The fitness function must provide a fitness for any possible individual, no matter how bad, and distinguish between any two individuals, no matter how close they are. The fitness function for this work compares energies and forces computed for a given set of atomic conformations using the evolving parameters with externally supplied energies and forces. Conformations for both near equilibrium and far from equilibrium configurations for very high and low energies were used

In general GA is not guaranteed to find a unique or even a satisfactory solution, but often works well in practice. JavaGenes uses many "GA parameters" (mutation rate, tournament size, etc.) that can affect performance and results of the search procedure. Choosing GA parameters is a non-trivial problem. We solve this by randomizing the choice of GA parameters in appro-

priate ranges in many parallel GA jobs. This eliminates a tedious human-directed search for good GA-parameters. Initially, 30-100 single-workstation GA runs with identical GA-parameters (except the random number seed) for each job were run with populations varying between 1000-3000. The GA-parameters that worked for one search (say, Si dimers in the fitness function) would fail in a similar search for a different system (say, larger Si clusters). The alternate technique of using a thousand trajectories with randomized GA-parameters and smaller populations (100-200) worked very well for all the systems attempted.

Using the evolutionary algorithm described above to automate force-field parameterization, we were able to reproduce the Stillinnger-Weber parameterization for Si [14], and generate parameters new force field functions for Si and Ge for a variety of nanotechnology applications. The new Si parameters matched the energetics of small Si clusters much better than Stillinger-Weber, and the new Ge parameters are the first available for the Stillinger-Weber functional form.

3 Fault Recovery on FPGAs

Most evolutionary approaches to fault recovery in FPGAs focus on evolving alternative logic configurations as opposed to evolving the intra-cell routing. Since the majority of transistors in a typical FPGA are dedicated to interconnect, nearly 80% according to one estimate, evolutionary fault-recovery systems should benefit by accommodating routing. In this section, we describe an evolutionary fault-recovery system employing a genetic representation that takes into account both logic and routing configurations. Experiments were run using a software model of the Xilinx Virtex FPGA. We report that using four Virtex combinational logic blocks, we were able to evolve a 100% accurate quadrature decoder finite state machine in the presence of a stuck-at-zero fault.

3.1 Approach

Bitstring representations are a natural choice for FPGA applications, and many times the raw configuration string can be used as the representation. In our case, we chose a bitstring representation mainly out of convenience in programming. Since we knew that only a handful of CLBs would be evolved, our bitstrings would be at most 1000 bits long. We acknowledge that this approach would likely suffer as more CLBs were utilized and the corresponding bitstring enlarged to thousands of bits.

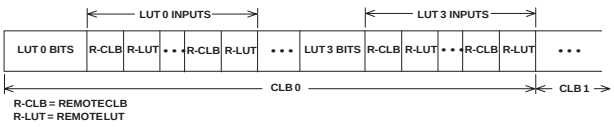

Figure 4: Genetic representation used showing logic fields and routing fields.

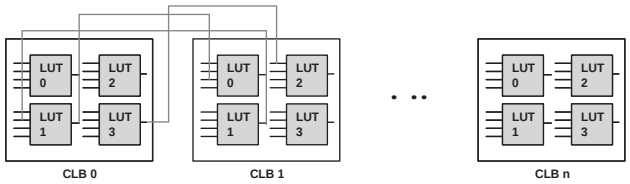

Figure 5: Example of routing among CLBs.

The representation is shown in Figure 4. This scheme is comprised of multiple 128-bit fields, one for each CLB. Within each CLB field are a number of sub-fields that specify each of the LUT bits and remote connections. There are 16 bits that specify the contents of each LUT. Each LUT has four inputs, and since each of these inputs can be connected to other LUT outputs, the remote CLB/LUT requires addressing bits. Since our system will be comprised of four CLBs, we need only two bits to specify the remote CLB, and another two bits to specify the particular LUT within the CLB. This pattern of sub-fields continues for each LUT until all the LUTs in the CLB are accounted for. An illustration of the CLBs, LUTs and sample routing is shown in Figure 5.

3.2 Experiments and Results

The quadrature decoder [1] was selected as an initial case study for testing and refinement of our evolutionary recovery strategy. It represents a NASA application of manageable size that is appropriate for tuning of the GA. Quadrature decoders provide a means of counting objects passed back and forth through two beams of light, or alternatively determining the angular displacement and direction of rotation of an encoder wheel turning about its axis. A quadrature decoder that determines the direction of rotation of a shaft is shown in Figure 6.

3.3 Experimental Setup and Results

The software system used is depicted in Figure 7. The entire system is implemented in software. The GA software is ECJ, a Java-based evolutionary computation and genetic programming system by Sean Luke of

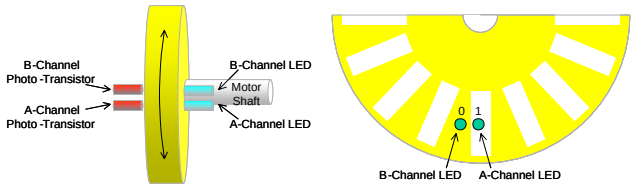

Figure 6: Rotating shaft application for a quadrature decoder.

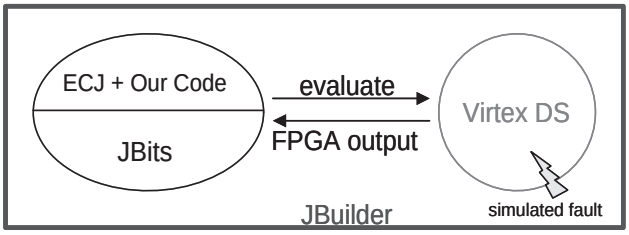

Figure 7: Software system for FPGA fault recovery.

George Mason University. ECJ is augmented by our code for tasks like decoding individuals and calculating fitness. The GA sits on top of Xilinx Corporation's JBits software [7], a set of Java classes which provide an Application Programming Interface to access the Xilinx FPGA bitstream. Xilinx's Virtex DS software, which simulates the operation of Virtex devices, is used to test candidate solutions. Borland's JBuilder Java environment is used for development and to run the system, though Sun Microsystem's Java virtual machine is used beneath JBuilder.

To evaluate the fitness of an individual, an input stream of 500 bit pairs is used. These inputs attempt to fully exercise the evolving finite state machines. The output stream consists of 510 bits sampled across all four CLBs. Ten bits are added to allow for delays in the evolved FSMs. This gives ten output stream windows of length 500, with each output stream shifted by 1-bit from the next. Sampling across all the CLBs allows the GA to maximum flexibility in building the FSM. Thus, fitness is expressed as:

$$F = \max_{i=1,4;j=0,9} (\mathrm{CLB}_i^j)$$

where CLB_i^j represents the number of correct output bits from the ith CLB shifted by j clock ticks. The fitness is simply the highest number of correct output bits seen across all of the CLBs and across the ten output windows. The best score is 500, and the worst score is 0.

Ten experimental runs were conducted using smaller input bitstreams of 100 bit pairs. These were found to evolve finite state machines that were tuned to the test cases, but not robust when interrogated with out of sample input test streams. Two runs were conducted using 500 bit pairs and one these runs was able to evolve a 100% accurate quadrature decoder finite state machine in the presence of an induced fault. The best evolved configuration was found in generation 623 and is shown in Figure 8. Two of the 16 LUTs went unused which is not surprising given that the FSM can be implemented with about 10 LUTs. The GA exploits the induced fault to its advantage because if you remove the fault in the evolved solution, it no longer functions correctly – it achieves an accuracy of only 93.8%. Also, note that the input LUTs had mostly zeros in their tables. This is because we fix most of those bits to zero in the genome since they do not affect the LUT's function. However, the "corner" bits of each of those input LUTs are involved in processing the input, and therefore, are evolved.

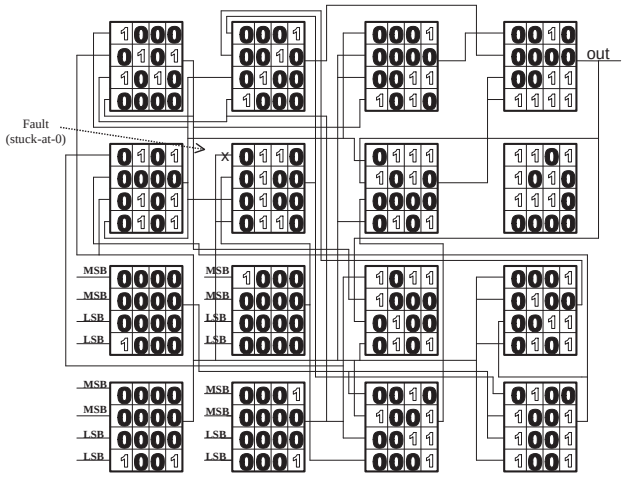

Figure 8: Evolved configuration showing routing, LUT contents, and simulated fault. Inputs are on the lines labeled MSB and LSB, referring to the least/most significant bit of the input. Wires that are shown crossing perpendicularly (eg, +) are unconnected – only wires that have ⊤ junctions are connected.

At the time of this writing, we have started running similar experiments using the actual hardware FPGA in the evolutionary loop (see Figure 9). The initial runs look promising. We have been able to evolve perfect quadrature decoders in the presence of 20 injected stuck-at faults in less than 5 minutes on a board clocked at 1 MHz. Even more encouraging is that we may be able to complete an entire evolutionary run in less than a minute. Our current system spends 90% of its time in software routing operations which we believe can be drastically reduced. On the hardware side, we have the ability to clock our board at much higher clock

Figure 9: FPGA Fault Recovery demonstration system.

frequencies.

4 Analog Circuit Design

In this section we outline a method of evolving analog electronic circuits using a linear representation and a simple unfolding technique [11]. While this representation excludes a large number of circuit topologies, it is capable of constructing many of the useful topologies seen in hand-designed circuits. Our system allows circuit size, circuit topology, and device values to be evolved. Using a parallel genetic algorithm we present initial results of our system as applied to two analog filter design problems. The modest computational requirements of our system suggest that the ability to evolve complex analog circuit representations in software is becoming more approachable on a single engineering workstation.

4.1 Approach

Circuits are represented in the genetic algorithm as a list of bytecodes which are interpreted during a simple unfolding process. A fixed number of bytecodes represent each component as follows: the first is the opcode, and the next three represent the component value. Component value encoding is discussed first.

Using three bytes allows the component values to take on one of 256^3 values, a sufficiently fine-grained resolution. The raw numerical value of these bytes was then scaled into a reasonable range, depending on the type of component. Resistor values were scaled sigmoidally between 1 and 100K ohms using $1/(1 + \exp(-1.4(10x-8)))$ so that roughly 75% of the resistor values were biased to be less than 10K ohms. Capacitor values were scaled between approximately 10 pF and 200 μF and inductors between roughly 0.1 mH and 1.5 H.

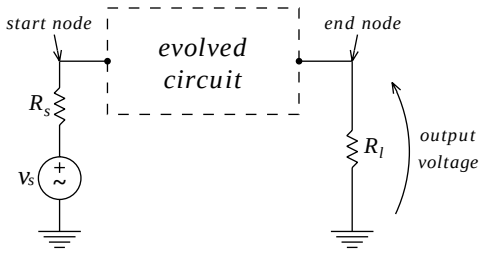

Figure 10: Artificially evolved circuit is located between fixed input and output terminals (v_s is an ideal ac voltage source, R_s is the source resistance, R_l is the load resistance).

The opcode is an instruction to execute during circuit construction. In the current design of our system, we use only "component placement" opcodes which accomplish placement of resistors, capacitors, and inductors. The five basic opcode types are: x-move-to-new, x-cast-to-previous, x-cast-to-ground, x-cast-input, x-cast-to-output, where x can be replaced by R (resistor), C (capacitor), or L (inductor). In a circuit design problem involving only inductors and capacitors (an LC circuit), ten opcodes would be available to construct circuits (five for capacitors and five for inductors).

The circuit is constructed between fixed input and output terminals as shown in Fig. 10. An ideal AC input voltage source v_s is connected to ground and to a source resistor R_s. The circuit's output voltage taken across a load resistor R_l.

To construct the circuit, a "current node" register (abbreviated CN; with "current" used in the sense of present, not electrical current) is used and initialized to the circuit's input node. The unfolding process then proceeds to interpret each opcode and associated component values, updating the CN register if necessary. The x-move-to-new opcode places one end of component x at the current node (specified by the CN register) and the other at a newly-created node. The CN register is then assigned the value of the newly-created node. The "x-cast-to-" opcodes place one end of component x at the current node and the other at either the ground, input, output, or previously-created node. After executing these opcodes, the CN register remains unchanged. The meanings of each opcode are summarized in Table 1. All five opcode types place components into the circuit, although they could be designed to do other actions as well, e.g., move without placement.

The list of bytecodes is a variable-length list (the

Opcode	Destination Node	CN Register
x-move-to-new	new-node	new-node
x-cast-to-previous	previous node	unchanged
x-cast-to-ground	ground node	unchanged
x-cast-to-input	input node	unchanged
x-cast-to-output	output node	unchanged

Table 1: Summary of opcode types used in current system. x denotes a resistor, capacitor, or inductor.

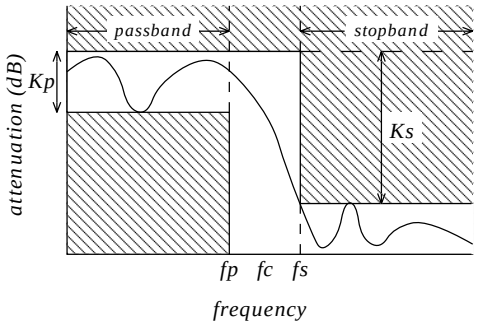

Figure 11: Low-pass filter terminology and specifications. The crosshatched regions represent out-of-specification areas. An example frequency response curve that meets specifications is shown.

length is evolved by the GA). Thus, circuits of various sizes are constructed. When the decoding process reaches the last component to place in the circuit, we arbitrarily chose to have the last node (value in CN) connected to the output terminal by a wire. By doing so, we eliminate unconnected branches.

4.2 Experiments and Results

The evolved circuit we present below is a low-pass filter. A low-pass filter is a circuit the allows low frequencies to pass through it, but stops high frequencies from doing so. In other words, it "filters out" frequencies above a specified frequency. The unshaded area in Fig. 11 depicts the region of operation for low-pass filters. Below the frequency f_p the input signal is passed to the output, potentially reduced (attenuated) by K_p decibels (dB). This region is known as the passband. Above the frequency f_s, the input signal is markedly decreased by K_s decibels. As labeled, this region is called the stopband. Between the passband and stopband the frequency response curve transitions from low to high attenuation. The parameter located in this region, f_c, is known as the cutoff frequency.

The low pass filter chosen was a circuit that can be built using a 3rd-order Butterworth filter. The specifications are as follows:

$f_p = 925$ Hz $\quad K_p = 3.0103$ dB
$f_s = 3200$ Hz $\quad K_s = 22$ dB

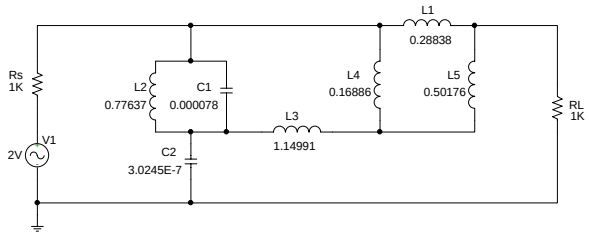

Figure 12: Evolved 3rd-order Butterworth low-pass filter (units are ohms, farads, and henries).

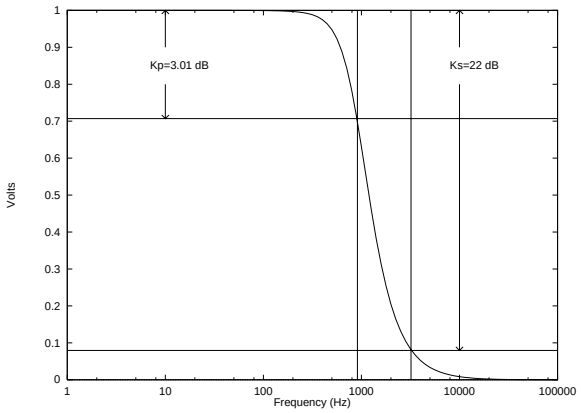

Figure 13: Frequency response curve for evolved 3rd-order Butterworth low-pass filter. Attenuation specifications are also shown. The frequency axis is a scaled logarithmically.

Such a filter design can be derived using a ladder structure and component values found in published tables. The GA was allowed to use capacitors and inductors during evolution, resulting in an LC low-pass filter. The evolved circuit that meets these specifications is shown in Fig. 12 and its frequency response is shown in Fig. 13. It was found in generation 22 of a GA run that lasted approximately four hours using six Sun Ultra workstations working in parallel.

5 EOS Satellite Scheduling

We hypothesize that evolutionary algorithms can effectively schedule coordinated fleets of Earth observing satellites. The constraints are complex and the bottlenecks are not well understood, a condition where evolutionary algorithms are often effective. This is,

Figure 14: Photograph of the Beowulf cluster.

in part, because evolutionary algorithms require only that one can represent solutions, modify solutions, and evaluate solution fitness. To test the hypothesis we have developed a representative set of problems, produced optimization software (in Java) to solve them, and run experiments comparing techniques. We've obtained initial results of a comparison of several evolutionary and other optimization techniques – namely the genetic algorithm, simulated annealing, squeaky wheel optimization, and stochastic hill climbing. We've also compared separate satellite vs. integrated scheduling of a two satellite constellation. While the results are not definitive, tests to date suggest that simulated annealing is the best search technique and integrated scheduling is superior. This work is described in [5] as well as a companion paper "Scheduling Earth Observing Satellites with Evolutionary Algorithms" in these proceedings.

6 Beowulf Cluster

A crucial component to most research in Evolvable Systems is the computational hardware. For the work reported above, our group uses an 80-cpu Beowulf Cluster as shown in Figure 14. The cluster uses a mixture of Intel and AMD cpus, standard office ethernet, and does not include disks at each node.

7 Discusssion

We have surveyed some of the research and development from the Evolvable Systems Group at NASA Ames Research Center. While most applications are currently in the proof-of-concept stage, we are investigating a mission insertion opportunity on one of our antenna designs. If successful, this would be one of the first fielded evolvable hardware applications.

References

[1] Agilent Technologies, Quadrature Decoder/Counter Interface ICs, Data Sheet HCTL-2020PLC.

[2] Al Globus, Charles Bauschlicher, Jie Han, Richard Jaffe, Creon Levit, Deepak Srivastava,"Machine Phase Fullerene Nanotechnology," Nanotechnology, 9, number 2, September 1998, pp. 192-199.

[3] A. Globus, J. Lawton, and T. Wipke, "Automatic molecular design using evolutionary techniques," Nanotechnology, Volume 10, Number 3, September 1999, pp. 290-299.

[4] "Evolving Molecular Force Field Parameters for Si and Ge," A. Globus, E. Ricks, M. Menon, D. Srivastava Proc. of the 2003 Nanotechnology Conf. Trade Show, February 23-27, 2003, San Francisco, California, USA.

[5] A. Globus, J. Crawford, J. Lohn, and R. Morris, "Scheduling Earth Observing Fleets Using Evolutionary Algorithms: Problem Description and Approach," Proceedings of the 3rd Intl. NASA Wkshp. on Planning and Scheduling for Space.

[6] "JavaGenes: Evolving Molecular Force Field Parameters with Genetic Algorithm," Al Globus, Madhu Menon, and Deepak Srivastava, Computer Modeling in Engineering and Science, vol. 3, no. 5, pp. 557-574, 2002.

[7] S. Guccione, D. Levi, P. Sundararajan, "JBits: A Java-based Interface for Reconfigurable Computing," 2nd Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD).

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

[9] D.S. Linden. "Wire Antennas Optimized in the Presence of Satellite Structures using Genetic Algorithms." IEEE Aerospace Conference, April 2000.

[10] D.S. Linden, "Automated Design and Optimization of Wire Antennas using Genetic Algorithms." Ph.D. Thesis, MIT, September 1997.

[11] J.D. Lohn, S.P. Colombano, "Automated Analog Circuit Synthesis using a Linear Representation," *Proc. of the Second Int'l Conf on Evolvable Systems: From Biology to Hardware*, Springer-Verlag, Berlin, 1998, pp. 125-133.

[12] J.D. Lohn, W.F. Kraus, D.S. Linden, "Evolutionary Optimization of a Quadrifilar Helical Antenna," Proc. of the IEEE AP-S International Symposium and USNC/URSI National Radio Science Meeting, June, 2002.

[13] Electromagnetic Optimization by Genetic Algorithms. Y. Rahmat-Samii and E. Michielssen, eds., Wiley, 1999.

[14] F. H. Stillinger, T. A. Weber, Dynamical Branching during Fluorination of the Dimerized Si(100) Surface: A Molecular Dynamic Study, Journal of Chemical Physics, 92(10), pages 6239-6245, May 1990.