J.D. Lohn, G.L. Haith, S.P. Colombano, D. Stassinopoulos, ``A Comparison of Dynamic Fitness Schedules for Evolutionary Design of Amplifiers," Proc. of the First NASA/DoD Workshop on Evolvable Hardware, Pasadena, CA, IEEE Computer Society Press, 1999, pp. 87-92.

A Comparison of Dynamic Fitness Schedules for Evolutionary Design of Amplifiers

Jason D. Lohn
Caelum Research Corp.
NASA Ames Research Center
Moffett Field, CA 94035-1000
jlohn@arc.nasa.gov

Gary L. Haith
Recom Technologies Corp.
NASA Ames Research Center
Moffett Field, CA 94035-1000
haith@ptolemy.arc.nasa.gov

Silvano P. Colombano
Computational Sciences Div.
NASA Ames Research Center
Moffett Field, CA 94035-1000
scolombano@mail.arc.nasa.gov

Dimitris Stassinopoulos
Computational Sciences Div.
NASA Ames Research Center
Moffett Field, CA 94035-1000
stassi@ptolemy.arc.nasa.gov

Abstract

High-level analog circuit design is a complex problem domain in which evolutionary search has recently produced encouraging results. However, little is known about how to best structure evolution for these tasks. The choices of circuit representation, fitness evaluation technique, and genetic operators clearly have a profound effect on the search process. In this paper, we examine fitness evaluation by comparing the effectiveness of four fitness schedules. Three fitness schedules are dynamic – the evaluation function changes over the course of the run, and one is static. Coevolutionary search is included, and we present a method of evaluating the problem population that is conducive to multiobjective optimization. Twenty-five runs of an analog amplifier design task using each fitness schedule are presented. The results indicate that solution quality is highest with static and coevolving fitness schedules as compared to the other two dynamic schedules. We discuss these results and offer two possible explanations for the observed behavior: retention of useful information, and alignment of problem difficulty with circuit proficiency.

1 Introduction

High-level circuit design is concerned with producing designs capable of achieving desired functions, while ignoring most implementation details. Low-level design activities are then used to factor in implementation considerations (e.g., placement and wiring of the devices) for realization of the target circuit. Electronic design automation (EDA) software has traditionally focused on low-level design activities: in some cases it hastens the design process or optimizes aspects of it, and in others, it is a prerequisite in order to deal with circuits having millions of components (e.g., memory) or vastly complex circuitry (e.g., microprocessors). EDA tools for high-level circuit design are less common – most of this work is accomplished by experienced engineers. Automated high-level circuit design techniques have started to appear in recent years, representing a variety of approaches (e.g., [2], [9], [6]). One approach which appears promising and is the subject of the work reported below, is the application of evolutionary search to high-level analog circuit design [5, 7, 11].

One of the goals of work in evolvable hardware is to automatically synthesize circuits that are better than those produced by an expert circuit designer. While recent results provide hope that we may achieve this goal, more research is needed. Thus it makes sense to investigate how to best structure evolutionary search for these tasks. The main ingredients – circuit representation, fitness evaluation technique, and genetic operators – each have a profound effect on the search process. We focus on fitness evaluation in the work reported below. Specifically, using an analog amplifier design task, we compare the effectiveness of four fitness schedules:

- *static schedule* - a single fitness function is used to evaluate all individuals throughout the run,

- *fixed schedule* - a pre-determined schedule of fitness functions whereby difficulty is increased at regular intervals,
- *adaptive schedule* - fitness function difficulty is increased only when circuit performance is commensurate,
- *coevolutionary schedule* - a population of constraints ("problems") is introduced consisting of individuals that embody a level of difficulty. These problems then coevolve with the circuits ("solutions") such that the difficulty increases as the circuits gain greater proficiency.

The goal of the design task was to generate a circuit that provides strong and constant amplification (gain) with low power dissipation and dc bias. This problem has the following desirable properties: amplifiers are practical circuits, optimization of multiple constraints makes them difficult to design, amplifier design is a relatively well explored design space, and it can be made more complex and difficult by adding further constraints and making the constraints more stringent.

The remainder of this paper is organized as follows. Section 2 provides more detail on analog circuit design and the specific design task used in the experiments. Section 3 elaborates on the fitness schedules. The experimental setup and results are presented in Sections 4 and 5, and we conclude in Section 6 with a discussion of the results.

2 The Design Task

The amplifier design task chosen was the inverting operational amplifier. Such a circuit has found wide application and is considered one of the workhorses of analog circuit design. Figure 1 shows the symbol and connections for an ideal inverting amplifier. This circuit generates an output voltage (v_o) that consists of the input voltage (v_i) multiplied by a gain factor, A. Voltage gain is thus equivalent to v_o/v_i. It is common to express gain values in decibels (dB) using $20\log_{10}(A)$. Amplifiers may be either inverting or non-inverting, where an inverted output signal has a 180° phase shift compared to the input. The dc gain of the amplifier refers to the gain when only constant voltage/current sources are applied. The linearity of the gain is the degree to which the gain remains constant across input voltages: ideally the voltage transfer characteristic (v_o vs. v_i) should be linear. The dc component that shifts the entire signal up or down is called the dc bias of the circuit. Power dissipation is the amount of power used

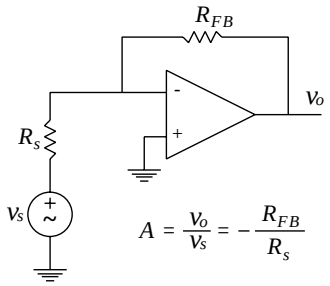

Figure 1: Ideal inverting amplifier showing how gain is set by the ratio of the feedback to source resistor.

by the circuit and is indicative of the amounts of current flowing in the circuit. For simple amplifiers, there are publications available that catalog many designs. Since there are numerous parameters in amplifier design (e.g., input/output impedance, power dissipation, distortion, common-mode rejection, power supply rejection), the design task can become quite challenging and typically requires an experienced designer. For the amplifier design experiments below, we take into account four objectives: dc gain, linearity of gain, dc bias, and power dissipation.

3 Fitness Schedules

The four fitness schedules mentioned above are discussed below in greater detail.

3.1 Static Fitness Schedule

The static fitness schedule is simply the standard evaluation technique in genetic algorithms [4]: a single fitness function is used to evaluate all individuals throughout the run. The fitness function used is similar to those described in [5, 7]. Briefly, it is a sum of normalized error values, where the errors are the shortfalls from the desired objectives: dc gain, dc bias, power dissipation, and the linearity of the dc gain. The gain is the slope of the dc transfer characteristic (i.e., the output voltages when the input voltage is swept across five input voltages). The slope, m, is calculated by using the endpoints of the transfer characteristic. The linearity of the gain is computed as $|m-m_l|+|m-m_r|$, where m_l is the slope of the line segment formed by the two leftmost output voltages and m_r is analogous for the two rightmost output voltages. The dc bias is simply v_o when $v_i = 0$ volts, and power dissipation is the amount of power consumed during circuit operation. The gain objective was 60.0 dB, the bias and power dissipation objectives were 1.0 volt and 1.0 watt, respectively, and

the linearity objective was 10.0. These values were chosen based on our previous work [8]: they represent a moderately difficult design task that we knew to be solvable.

3.2 Fixed Fitness Schedule

The fixed fitness schedule is a pre-determined schedule of fitness function modifications. As used in the experiments below, the difficulty-level of the fitness function is increased every 50 generations. With a total of 5000 generations, this allowed for a total of 100 "difficulty steps." Each of the fitness functions used over the course of the run are of the same form as the fitness function used in the static schedule above. Writing our gain, bias, power, and linearity objectives as a *target vector*, $\langle G, B, P, L\rangle$, we specified that the difficulty level begins at $\langle 1.0, 10.0, 10.0, 1000.0\rangle$ (easiest) and ends at $\langle 60.0, 1.0, 1.0, 10.0\rangle$ (most difficult). The increases in difficulty are then evenly divided over the 100 steps, per objective. This is admittedly an arbitrary schedule, but that is an inherent property of a fixed schedule - it is subject to the biases of the implementor. Such biases can be advantageous if knowledge of the fitness landscape is known *a priori*, and potentially disadvantageous otherwise.

3.3 Adaptive Fitness Schedule

The adaptive fitness schedule is identical to the fixed schedule described above except in the following regard: difficulty is incremented "on-demand," whenever the current difficulty is solved by at least one circuit in the population. As in the fixed schedule case, 100 difficulty steps are provided for. If a circuit solves the 100th fitness function before 5000 generations, it has successfully found a compliant circuit, and the run halts. On the other hand, if 5000 generations elapse and a compliant circuit is not found, the run halts at whatever difficulty level it has reached.

3.4 Coevolving Fitness Schedule

The main difference between the coevolving fitness schedule and the other dynamic schedules is the introduction of a second population consisting of target vectors (tv). The first population of circuits remains the same as in the other fitness schedules. The target vector population consists of individuals that specify problem difficulty. As described above, target vectors are denoted $\langle G, B, P, L\rangle$, representing gain, bias, power dissipation, and gain linearity, respectively. The individual targets are threshold values – a target is "solved" if a circuit's performance equals or surpasses (either above or below, as appropriate) the threshold specified. For example, $\langle 63.0, 0.6, 0.8, 9.5\rangle$ solves $\langle 60.0, 1.0, 1.0, 10.0\rangle$, but $\langle 58.0, 0.6, 1.2, 18.0\rangle$ does not. As with the other fitness schedules, the ideal target vector used was $\langle 60.0, 1.0, 1.0, 10.0\rangle$. The gain target is satisfied if a circuit's gain was 60.0 decibels or greater. The three remaining targets were satisfied if the circuit's performance is less than or equal to the target values.

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.

Fitness of individual circuits in the main population was computed as follows. Circuit i "plays" each target vector in the second population and a score, s_i, is computed:

$$s_i = \sum_{j \in \widehat{\mathrm{tv}_i}} \frac{1}{\begin{array}{c}\text{total \# circuits}\\ \text{that solve } \mathrm{tv}_j\end{array}}$$

where $\widehat{\mathrm{tv}_i}$ is the set of target vector indexes such that circuit i solves tv_j. Note that the denominator in the above fraction is guaranteed to be greater than or equal to one due to the restriction on j. Then s_i is normalized linearly between its upper and lower bounds such that 0.0 is the best score and 1.0 the worst:

$$F(\mathrm{circuit}_i) = 1.0 - s_i/M_2$$

where M_2 is the size of the target vector population. The effect of s is to reward circuits that solve the more difficult target vectors. A target vector has the greatest difficulty level when exactly one circuit can solve it. If many circuits can solve a particular target vector, the fitness contribution in s is shared among the circuits [10].

Fitness of an individual target vector is computed as follows. Let x_j denote the number of circuits that solve tv_j, and M_1 be the circuit population size. The fitness is essentially x_j, scaled and normalized, with a tractability constraint:

$$F(\mathrm{tv}_j) = \begin{cases} 1.0 & x_j = 0 \\ \frac{1}{(M_1 - 1)}(x_j - 1.0) & x_j \geq 1 \end{cases}$$

The tractability constraint gives a target vector a score of 1.0 (the "worst" score) when no circuits can solve it. This puts pressure on the target vector population to pose difficult, yet solvable problems to the circuit population.

4 Experimental Setup

Using the four fitness schedules described above, 25 runs using each schedule were made resulting in a total of 100 runs. The same pseudo-random number generator seed was used across each set of four distinct fitness schedules so that the generation zero individuals would be identical. Common to each run were the following parameter settings: population size was 600, crossover rate was 80%, mutation rate was 5%. For the coevolution runs,the target vector population used the following parameters: population size was 600, crossover rate was 80%, mutation rate was 50%. Because crossover points were chosen between target vector values, this mutation rate was set high to encourage new values to appear in the population, not just those produced in generation 0.

Evolution of amplifier designs was accomplished using the system described in [7]. Briefly, circuits are represented as lists of circuit-construction instructions that program an automaton to design a circuit. Resistors, capacitors, and bipolar junction transistors were the allowed components. The method of incorporating transistors is described in [8]. Circuits were required to contain at least 10 components up to a maximum of 150.

5 Experimental Results

To assess the quality of each fitness schedule, we examined the highest fitness circuits from each run. The performance of these circuits is quantified in corresponding *output vectors* which, like target vectors, specify gain, bias, power dissipation, and linearity values. Table 1 gives the mean values of individual objectives across output vectors for each fitness schedule. The data suggest that static and coevolving fitness schedules performed better than fixed and adaptive schedules. Another way of measuring the quality of the fitness schedules is to look at the number of objectives solved in each run (assuming each of the four objectives is of equal importance). Table 2 shows the mean and standard deviation for the number of objectives solved for each schedule.

Here the relationship among the schedules is clearer: static and coevolving fitness schedules performed nearly the same and did better than the performance of the fixed and adaptive schedules. A two-tailed t-Test showed that the static and coevolving means are not significantly different from each other, and are significantly different ($p < 0.02$) from the fixed and adaptive means.

One of the motivations behind using coevolutionary search is the notion that the problem difficulty is adjusted automatically, rather than having to manually specify it. To get a sense of how coevolution accomplished this, Figure 2 shows four plots (one for each target objective), each containing 25 curves (fitted using a fourth-order polynomial). The plots show how the values of the best target vectors found in each generation fluctuated during the run. The thick curve represents the run that found a compliant circuit (i.e., it solved $\langle 60.0, 1.0, 1.0, 10.0 \rangle$).

What is most striking is the way coevolution, within the first few generations, reduced the demands for gain performance because it was the most difficult criterion to meet. Just as rapidly, the other three objectives were made more demanding because they were relatively easy to satisfy. Then as the circuit population scored better in gain, it did so at the expense of power and linearity: both power and linearity are seen peaking near generations 1000-2000.

fitness schedule	gain [dB]	bias [volts]	power [watts]	linearity [unitless]
static	44.47	0.35	0.69	49.28
fixed	47.59	0.64	1.21	96.63
adaptive	54.13	1.23	1.96	340.74
coevolving	46.71	0.15	0.41	189.75

Table 1: Mean values from the performance of the best circuits found under 25 runs of each fitness schedule. The ideal target vector was $\langle 60.0, 1.0, 1.0, 10.0 \rangle$.

fitness schedule	mean	std. dev.
static	2.12*	0.67
fixed	1.48	1.12
adaptive	1.16	1.18
coevolving	2.08*	0.49

Table 2: Mean and standard deviation for the number of objectives solved for 25 runs of each fitness schedule. Means marked with asterisks (*) are not significantly different from each other, and are significantly different ($p < 0.02$) from those means without asterisks.

6 Discussion

From the results it is seen that static and coevolutionary fitness schedules outperformed the fixed and adaptive schedules. Although is not completely clear why this happened, we can offer potential advantages of the

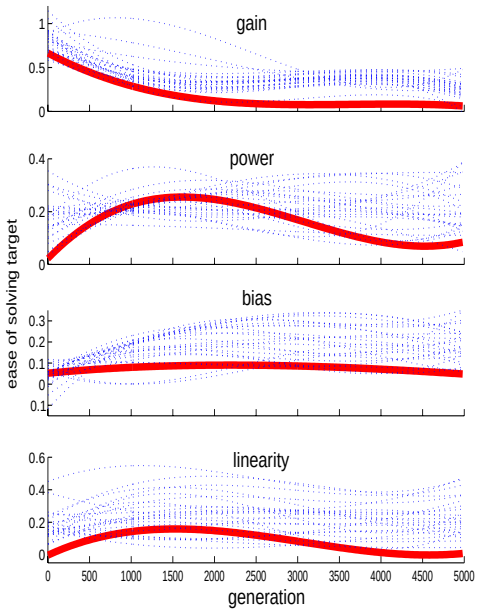

Figure 2: Highest fitness target vector values over the course of all coevolution runs. The y-axes represent the difficulty of the objective with 0.0 being the target (or most difficult) value, and 1.0 being the easiest objective value. The thick curves represent the run that found a compliant circuit. Curves were fitted using a fourth-order polynomial, and therefore sometimes appear above 1.0 and below 0.0.

static and coevolutionary schedules relative to the fixed and adaptive schedules. First, because a static fitness function induces a fitness landscape that never changes over the course of evolution there is never the possibility of getting "thrown off" a gradient (as would be the case if the fitness function changed). Second, we designed coevolution so that it would keep the level of problem difficulty near the leading edge of circuit proficiency. Developmental theory suggests (e.g., [1]) that keeping task difficulty in line with solution performance aids learning. Third, the fixed and adaptive schedules are potentially "handicapped" by the somewhat arbitrary choice of manually-crafted schedules.

Dynamic fitness schedules can help evolutionary search because they encourage the population of circuits to follow potentially better trajectories through the solution space. Such trajectories could guide evolution in many ways, for example they could amplify weak gradients in the fitness landscape, "steer around" meta-stable solution states [10], and usefully decompose or simplify the problem by providing partial reinforcement for intermediate solutions [3]. As an illustration, an amplifier made up of a single wire has excellent performance in terms of bias, linearity and power dissipation, but has zero gain. Adding some components to the circuit might increase the gain, but only at the cost of a dip in performance on the other three criteria. Thus, if evolved with a static fitness schedule (assuming equally-weighted objectives), the single wire presents evolutionary search with a meta-stable state that is highly attractive and potentially quite difficult to escape. In contrast, the fixed fitness schedule in the present amplifier design task encourages all of the performance objectives (gain, power dissipation, bias, and linearity) to be solved in parallel by evolution. Likewise, coevolution tends to work on gain early in evolution and to scale back the requirements on bias, power and linearity until circuits are performing fairly well on gain.

One issue affecting our results is the question of parameter sensitivity. Although parameters were kept the same as possible across all runs, there are undoubtedly parameter changes that would strongly affect our results. Examples include the size of the difficulty steps in the fixed and adaptive schedules, and form of the coevolving fitness calculation.

Given the relatively good performance of coevolution, we feel that our method of using target vectors neatly and systematically breaks down multiobjective optimization problems and may perform well in other such problems. One of the subjects for further study is to see how well this technique scales up to optimization problems having many more objectives.

In conclusion, static and coevolving fitness evaluations did relatively well in our amplifier design task. Based on our previous work in evolving amplifier designs, we suspected that the static technique would be able to solve this design task. We find it very encouraging that coevolution performed on par with static fitness schedules and intend to pursue coevolutionary search in future circuit design tasks.

References

[1] J.L. Elman, *Incremental Learning, or the Importance of Starting Small,* Tech. Rept. 9101, Center for Research in Language, University of California, San Diego, CA, 1991.

[2] G. Gielen, W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits,* Boston, MA: Kluwer, 1991.

[3] G.L. Haith, S.P. Colombano, J.D. Lohn, D. Stassinopoulos, "Coevolution for Problem Simplification," *Proc. 1999 Genetic and Evolutionary*

Computation Conference, (GECCO-99), 1999, to appear.

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

[5] J.R. Koza, F.H. Bennett, D. Andre, M.A. Keane, F. Dunlap, "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 2, July, 1997, pp. 109–128.

[6] M.W. Kruiskamp, *Analog Design Automation using Genetic Algorithms and Polytopes,* Ph.D. Thesis, Dept. of Elect. Engr., Eindhoven University of Technology, Eindhoven, The Netherlands, 1996.

[7] 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.

[8] J.D. Lohn, S.P. Colombano, "A Circuit Representation Technique for Automated Circuit Design," *IEEE Trans. on Evolutionary Computation*, to appear.

[9] E.S. Ochotta, R.A. Rutenbar, L.R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," *IEEE Trans. Computer-Aided Design,* vol. 15, pp. 273–294, 1996.

[10] C.D. Rosin, R.K. Belew, *New Methods for Competitive Coevolution,* Tech. Rept. CS96-491, Department of Computer Science and Engineering, University of California, San Diego, 1996.

[11] R.S. Zebulum, M.A. Pacheco, M. Vellasco, "Comparison of Different Evolutionary Methodologies Applied to Electronic Filter Design," *1998 IEEE Int. Conf. on Evolutionary Computation*, Piscataway, NJ: IEEE Press, 1998, pp. 434–439.