Evolution of a Time-Optimal Fly-To Controller Circuit using Genetic Programming

John R. Koza
Computer Science Dept.
258 Gates Building
Stanford University
Stanford, California 94305-9020
koza@cs.stanford.edu
http://www-cs-faculty.stanford.edu/~koza/

Forrest H Bennett III
Visiting Scholar
Computer Science Dept.
Stanford University
Stanford, California 94305
forrest@evolute.com

Martin A. Keane
Martin Keane Inc.
5733 West Grover
Chicago, Illinois 60630
makeane@ix.netcom.com

David Andre
Computer Science Division
University of California
Berkeley, California
dandre@cs.berkeley.edu

ABSTRACT

Most problem-solving techniques used by engineers involve the introduction of analytical and mathematical representations and techniques that are entirely foreign to the problem at hand. Genetic programming offers the possibility of solving problems in a more direct way using the given ingredients of the problem. This idea is explored by considering the problem of designing an electrical controller to implement a solution to the time-optimal fly-to control problem.

1. Introduction

Suppose an engineer is faced with the problem of designing an electronic device to control the flight of an aircraft with a constrained turning radius such that the aircraft flies to an arbitrary destination point in minimal time.

The required end-product is an electronic device composed of a particular topological arrangement of transistors, diodes, power sources, and resistors of various values. The required behavior is expressed in terms of time. However, even though the original given ingredients of the stated problem are electronic components and time, a practicing engineer would not ordinarily directly work with these original given ingredients in solving the problem. Instead, the engineer would typically inject into the problem-solving process a large number of additional technologies that are entirely foreign to the problem at hand and then try to solve the problem using those technologies. The engineer's approach might well involve steps such as the following three.

First, the engineer might introduce mathematics into the problem. His goal would be to create a mathematical model that extracts key features of the problem and represents them as mathematical objects that can be manipulated using mathematical methods. For example, the engineer might draw a geometric diagram containing points representing the aircraft and its destination, circles reflecting the aircraft's constrained turning radius, and various vectors representing trajectories of the aircraft. The engineer might then use the diagram to write equations that capture certain key relationships of the problem.

Second the engineer might solve the mathematical problem created by his geometric and mathematical analysis. For the version of the fly-to problem involving an aircraft flying at constant speed and altitude, Clements (1990) found the Hamiltonian and La grange multipliers for this optimal control problem and then used the Pontryagin minimization theorem to find the aircraft's optimal control strategy.

Third, the engineer might introduce computer science techniques and electronic design techniques into the problem so that he could design the desired electronic circuit for controlling the flight of an aircraft to implement the solution to the mathematical problem.

If the engineer elected to implement the mathematical solution with a mixed analog-digital controller, he would use some programming language to write the computer program embodying the mathematical steps of the optimal control strategy. A compiler would then be used to translate the individual instructions of the programming language into executable machine code. This machine code would then be executed by a digital microprocessor that would perform, say, floating-point addition, subtraction, multiplication, and division on binary numbers in the fixed-size registers of the microprocessor. Finally, a digital-to-analog converter would be used to convert the output of the digital processor into the analog signal that will actually control the aircraft's heading.

Alternatively, if the engineer elected to implement the mathematical solution with a purely analog controller, he would forego the purely digital processing and instead go through the process of designing an analog circuit (effectively an analog computer) that would perform the mathematical computations with analog electrical signals for optimally controlling the aircraft's heading.

The design of analog circuits, in general, is difficult and has not proved to be amenable to automation. Part of the difficulty of the analog design process stems from the fact that the behavior of analog circuits is specified by a system of integro-differential equations (one equation for each node and loop in the circuit in accordance with Kirchhoff's node and loop laws). When the circuit contains only capacitors, inductors, and resistors, the equations in the system are "merely" linear integro-differential equations and mathematical techniques such as Laplace transforms can be used to solve the equations. When the circuit contains components such as transistors and diodes (which are present in most electronic circuits and, in any event, are necessary for the problem at hand), the equations are non-linear and vexatious. Even when it is possible to solve these equations for the behavior of a circuit, it is difficult to design (synthesize) a circuit with that particular behavior. In discussing the difficulty of designing analog circuits, Aaserud and Nielsen (1995) observe,

> "Analog designers are few and far between. In contrast to digital design, most of the analog circuits are still handcrafted by the experts or so-called 'zahs' of analog design. The design process is characterized by a combination of experience and intuition and requires a thorough knowledge of the process characteristics and the detailed specifications of the actual product.
>
> "Analog circuit design is known to be a knowledge-intensive, multiphase, iterative task, which usually stretches over a significant period of time and is performed by designers with a large portfolio of skills. It is therefore considered by many to be a form of art rather than a science."

Notice that regardless of whether the engineer elects to implement the solution as a mixed analog-digital circuit or a purely analog circuit, he must inject into the problem-solving process numerous representations and technologies that are entirely foreign to the problem at hand, including

- geometry,
- mathematical equations
- Hamiltonians,
- Lagrande multipliers, and
- Pontryagin's principle

plus either

- digital microprocessors,
- programming languages,
- compilers
- digital-to-analog converters

or plus

- integro-differential equations,
- Laplace transforms, and
- analog design techniques.

This paper discusses genetic programming as a problem-solving approach that works in a more direct way with the original given ingredients of the problem and, specifically, avoids or minimizes the introduction of additional representations and technology that are foreign to the problem at hand. This approach is applied to the problem of building an electrical controller device to discover the solution to the time-optimal fly-to control problem. The approach that is used herein could be directly applied to many other problem areas.

Section 2 presents the fly-to problem and its mathematical solution. Section 3 presents the preparatory steps for the solution to the fly-to problem using genetic programming. Section 4 presents the results.

2. The Fly-To Problem

In the fly-to problem (Clements 1990), the goal is to find a program for controlling the flight of an aircraft (flying at constant speed and altitude) such that the aircraft flies to an arbitrary destination point in minimal time.

The difficulty of this optimal control problem arises from the fact that the aircraft's nonzero minimum turning radius generally precludes flying directly to the destination. Like other non-trivial optimal control problems, this problem cannot be solved merely by repetitively executing a hill-climbing action that greedily improves the distance between the aircraft and the destination at every intermediate point along the aircraft's trajectory. Instead, such problems require executing various seemingly disadvantageous actions in order to later achieve the globally optimal result.

In figure 1, the aircraft is initially positioned at the origin (0, 0) of the coordinate system and is headed east (i.e., along the positive x axis). The aircraft files at a constant air speed, A. The aircraft's maximum turn angle $\Theta_{\max}$ defines a turning radius, R, for the aircraft. The two circles centered at (0, $+R$) and (0, $-R$) in the figure each have a radius equal to this turning radius. The aircraft's constant-altitude flight is controlled by the turn angle Θ, which controls the aircraft's change in heading. Angles are measured in radians counterclockwise from the positive x axis.

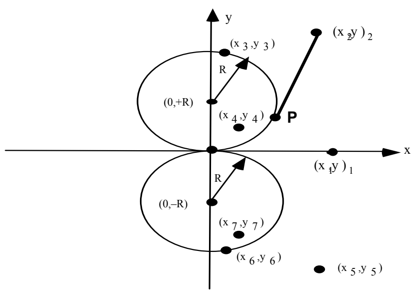

Figure 1 The first three cases of the fly-to problem.

Suppose the fly-to (target) point lies on the positive x axis, such as point (x_1, y_1) in the figure. In this first case, flying straight ahead (i.e., east) will bring the aircraft to the target in minimum time. Thus, the time-optimal control strategy for the aircraft is to fly straight ahead along the positive x axis (i.e., with $\Theta = 0$ radians). The aircraft's time-optimal trajectory will be the straight line between the origin and the target (x_1, y_1) along the positive x axis.

In the second case, the target point, such as (x_3, y_3), lies on the circumference of the upper circle of radius *R*. For this second case, the time-optimal control strategy is to turn with a heading equal to the maximum turn angle $\Theta = +\Theta_{\max}$ (i.e., initially northeast) The aircraft's time-optimal trajectory will be the portion of the circumference of the circle between the origin and the target (x_3, y_3). By symmetry, the minimum-time control strategy for any point, such as (x_6, y_6), on the circumference of the lower circle involves heading at the maximum turn angle of $\Theta = -\Theta_{\max}$ (i.e., initially southeast).

The remaining points can be classified into two additional cases based on whether they are outside or inside the circles. The strategies for these next two cases employ portions of the aircraft trajectories used by one or both of the above strategies.

In the third case, the target point is in the first and second quadrants but *outside* the upper circle. For a point such as (x_2, y_2), there is a unique first point **P** on the circumference of the upper circle such that the tangent line to the upper circle at point **P** passes through (x_2, y_2). For this third case, the minimum-time control strategy for the aircraft is to turn at the maximum angle $\Theta = +\Theta_{\max}$ until the aircraft reaches point **P** on the circumference of the upper circle and then to fly straight ahead (i.e., with an angle of $\Theta = 0$) to the target. The aircraft's time-optimal trajectory will be the portion of the circumference of the upper circle between the origin and **P** combined with the tangent line between **P** and the target (x_2, y_2). Note that this strategy works regardless of the location of (x_2, y_2) in the first or second quadrants and outside the upper circle (including, specifically, points in the second quadrant behind the aircraft). By symmetry, a similar strategy (of first turning southeast at the maximum angle of $\Theta = -\Theta_{\max}$ and then flying straight) works for target points *outside* the lower circle, such as (x_5, y_5). Points on the negative *x* axis (i.e., behind the aircraft) can be reached equally well in this manner (initially going around the circle in either direction).

In the fourth case, the target point is in the first or second quadrants but *inside* the upper circle. None of the points inside the upper circle can be reached by pursuing a hill-climbing action that greedily improves the distance between the aircraft and the target at every intermediate point along the aircraft's trajectory. Such points can only be reached by incurring a substantial temporary increasing of the distance between the aircraft and the target. Figure 2 shows, for a point such as (x_4, y_4), that there is a unique point **Q** to the east of the origin on the circumference of the lower circle such that a unique circle is tangent to the lower circle at point **Q** and passes through (x_4, y_4) of the upper circle. For this fourth case, the minimum-time control strategy for the aircraft is to turn at the maximum angle $\Theta = -\Theta_{\max}$ (i.e., initially turning southeast thereby increasing the distance between the aircraft and the target) until the aircraft reaches point **Q** and then to reverse directions and turn at the maximum angle $+\Theta_{\max}$ until the aircraft reaches the target. The aircraft's time-optimal trajectory will be the portion of the circumference of the lower circle between the origin and **Q** and the portion of the circumference of the new tangent circle between **Q** and the target. Note that this strategy works regardless of the location of (x_4, y_4), in the first or second quadrants and inside the upper circle (including, specifically, points behind the aircraft in the second quadrant). By symmetry, a similar strategy (of first turning northeast at the maximum angle of $\Theta = +\Theta_{\max}$ and then turning at the maximum angle of $\Theta = -\Theta_{\max}$) works for target points *inside* the lower circle, such as (x_7, y_7).

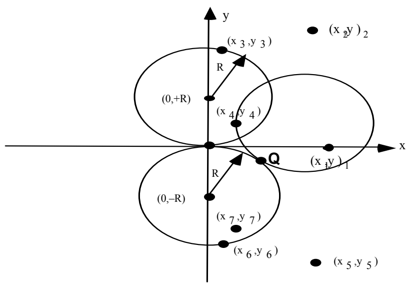

Figure 2 The fourth case of the fly-to problem.

The above optimal control strategies and trajectories were described as if the state of the system were the aircraft's changing position relative to the ground, namely (x_{ground}(t), y_{ground}(t)). If the control variable for the aircraft is the turn angle Θ, then the state-transition equations for the system are

$$x_{ground}(t+1) = x_{\text{ground}}(\text{t}) + A\ \ \Delta t\ \ \text{Cos}\ \Theta$$

$$y_{ground}(t+1) = y_{\text{ground}}(\text{t}) + A\ \ \Delta t\ \ \text{Sin}\ \Theta$$

The more natural viewpoint for an aircraft controller is the view from the aircraft. In this view, whenever the aircraft travels in a particular direction, the coordinate system is immediately adjusted so that the aircraft is repositioned to the origin (0, 0) of the coordinate system with the aircraft heading due east. In this view, the state of the system is the changing location of the target (fly-to) point.

Each of the above four optimal control strategies can be restated in accordance with this new view from the aircraft. For the first case, the strategy of flying straight ahead toward a target on the positive *x* axis has the effect of moving the target point west along the positive *x* axis until it reaches the aircraft. For the second case, the strategy of turning with a turn angle $\Theta_{\max}$ for a target point on the circumference of the upper circle has the effect of moving the target point around the circumference of the circle until it reaches the aircraft. For the third case, the strategy for a target point in the first or second quadrants and outside the upper circle has the effect of first rotating the target to the positive *x* axis and then moving the target west along the positive *x* axis until it reaches the aircraft. For the fourth case, the strategy for a target point inside the upper circle has the effect of first rotating the target away from the aircraft to the circumference and then moving the target point around the circumference until it reaches the aircraft.

As a specific example, consider an aircraft whose constant air speed, A, is 200 knots and whose maximum performance rate of turn corresponds to a turn angle of 0.197 radians. The aircraft is located at the origin of a coordinate system representing a world extending 4 nautical miles in each of the four directions. A total of 80 time steps of 0.001 hour each are used in simulating the trajectory of the aircraft. The total simulation time of 0.080 hours generously permits a trajectory equivalent to the aircraft flying twice across its world of 64 square nautical miles.

3. Preparatory Steps

We used genetic programming (an extension of the genetic algorithm described in Holland 1975) in which the population consists of computer programs of varying sizes and shapes as described in Koza 1992, 1994a, 1994b; Koza and Rice 1992; Kinnear 1994; Angeline and Kinnear 1996, and Koza, Goldberg, Fogel, and Riolo 1996).

We used the methods for designing analog electrical circuits with genetic programming described in Koza, Andre, Bennett, and Keane (1996), and Koza, Bennett, Andre, and Keane (1996a, 1996b, 1996c, 1996d, 1997). See also Gruau's innovative work on cellular encoding (1996).

Before applying genetic programming to a problem of circuit synthesis, the user must perform seven major preparatory steps, namely (1) identifying a suitable embryonic circuit, (2) determining the architecture of the overall circuit-constructing program trees, (3) identifying the terminals of the to-be-evolved programs, (4) identifying the primitive functions contained in the to-be-evolved programs, (5) creating the fitness measure, (6) choosing certain control parameters (notably population size and the maximum number of generations to be run), and (7) determining the termination criterion and method of result designation.

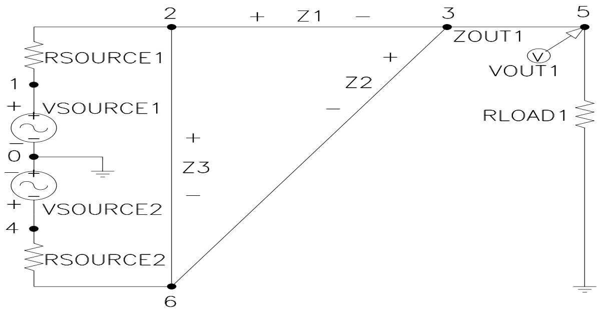

Figure 3 Two-input, one-output embryonic circuit.

The electrical circuit to solve the fly-to problem requires two analog inputs and one analog output. The two inputs consist of the current location ($x_{fly\text{-}to}$(t), $y_{fly\text{-}to}$(t)) of the target fly-to point in relation to an aircraft heading due east and positioned at the origin of the coordinate system. The output is interpreted by the wrapper (output interface) to be the aircraft's turn angle Θ. Figure 3 shows an embryonic circuit for a two-input, one-output circuit.

The number of automatically defined functions, if any, will emerge as a consequence of the evolutionary process using the architecture-altering operations (Koza 1994c). Since the embryonic circuit has three modifiable wires, there are three writing heads and three result-producing branches (RPB0, RPB1, RPB2) in each circuit-constructing program tree. Each program in the initial population of programs has a uniform architecture with no automatically defined functions (i.e., three result-producing branches).

For this problem, the function set, $\mathcal{F}_{\text{ccs}}$, for each construction-continuing subtree is

$\mathcal{F}_{\text{ccs}}$ = {R, SERIES, PSS, PSL, FLIP, NOP, NEW_T_GND_0, NEW_T_GND_1, NEW_T_POS_0, NEW_T_POS_1, NEW_T_NEG_0, NEW_T_NEG_1, PAIR_CONNECT_0, PAIR_CONNECT_1, Q_D_NPN, Q_D_PNP, Q_3_NPN0, ..., Q_3_NPN11, Q_3_PNP0, ..., Q_3_PNP11, Q_POS_COLL_NPN, Q_GND_EMIT_NPN, Q_NEG_EMIT_NPN, Q_GND_EMIT_PNP, Q_POS_EMIT_PNP, Q_NEG_COLL_PNP}

SPICE's default npn and pnp transistor model parameters were used.

Space does not permit a detailed description of each component-creating and connection-modifying function. For details, see Koza, Andre, Bennett, and Keane (1996), and Koza, Bennett, Andre, and Keane (1996a, 1996b, 1996c, 1996d, 1997).

The terminal set, $\mathcal{T}$ccs, for the construction-continuing subtree is

$\mathcal{T}$ccs = {END, SAFE_CUT}.

The function set, $\mathcal{F}$aps, for each arithmetic-performing subtree is,

$\mathcal{F}$aps = {+, -}.

The terminal set, $\mathcal{T}$aps, for each arithmetic-performing subtree consists of

$\mathcal{T}$aps = {$\leftarrow$},

where $\leftarrow$ represents floating-point random constants from –1.0 to +1.0.

The fitness of a controller was evaluated using 72 randomly chosen fitness cases each representing a different target (fly-to) point. Fitness was the sum, over the 72 fitness cases, of the fly-to times. A smaller sum is better. If the aircraft came within a capture radius of 0.28 nautical miles of its target point before the end of the 80 time steps allowed for a particular fitness case, the contribution to fitness for that fitness case was the actual fly-to time. However, if the aircraft failed to come within the capture radius during the 80 time steps, the contribution to fitness was 0.160 hours (i.e., double the worst possible time).

The calculation of fitness requires up to 5,760 executions (72 ∞ 80) of each individual program. The world of 64 square nautical miles was discretized into a 40 ∞ 40 grid of 1,600 discrete squares (with each square being 0.2 ∞ 0.2 nautical miles). A table was computed for each individual program in the population giving the value of the control variable Θ as if the target points were in the center of each of the 1,600 squares and the aircraft were at (0,0). When an individual program was executed for a particular one of the 80 time steps of a particular one of the 72 fitness cases, the state of the system was computed from the above state-transition equations using floating-point arithmetic. However, the value of the control variable, Θ, was obtained from the table as if the target were located in the center of its square. In addition, if the aircraft flew outside of its world of 64 square nautical miles or if the target point was moved out of the world relative to the aircraft's frame of reference, the simulation was terminated and that fitness case was assigned the penalty value of fitness of 0.160 hours.

For this problem, the voltage VOUT is probed at node 5. The SPICE simulator (Quarles et al. 1994) is requested to perform a nested DC sweep. The nested DC sweep provides a way to simulate the DC behavior of a circuit with two inputs. A nested DC sweep resembles a nested pair of FOR loops in a computer program in that both of the loops have a starting value for the voltage, an increment, and an ending value for the voltage. For each voltage value in the outer loop, the inner loop simulates the behavior of the circuit by stepping through its range of voltages. Specifically, the starting value for voltage is –4 volts, the step size is 0.2 volts, and the ending value is +4 volts. These values correspond to the dimensions of the aircraft's world of 64 square nautical mile extending 4 nautical miles in each of the four directions from the origin of a coordinate system (i.e., 1 volt equals 1 nautical mile).

When an individual program is executed, it produces a numeric value, x, which the wrapper transforms into the turn angle Θ. Figure 4 shows that the wrapper interprets the numeric value, x, returned by execution of an individual program by taking the output modulo 2π and clamping the absolute value to be less than or equal to $\Theta_{\max}$.

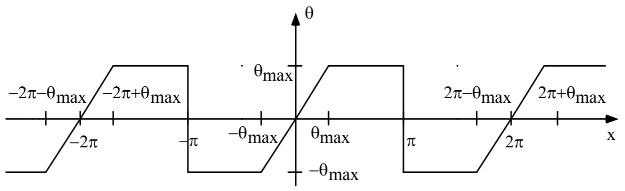

Figure 4 Wrapper for the fly-to problem.

The number of hits is defined as the number of fitness cases (from 0 to 72) for which the aircraft reaches its destination within 80 time steps (i.e., does not time out).

The population size, M, was 640,000. The architecture-altering operations are used sparingly on each generation. The percentage of operations on each generation after generation 5 were 86.5% one-offspring crossovers; 10% reproductions; 1% mutations; 1% branch duplications; 0.5% branch deletions; and 1% branch creations. Since we did not want to waste large amounts of computer time in early generations where only a few programs have any automatically defined functions at all, the percentage of operations on each generation before generation 6 was 78.0% one-offspring crossovers; 10% reproductions; 1% mutations; 5.0% branch duplications; 1% branch deletions; and 5.0% branch creations. The maximum size, H_{rpb}, for each of the three result-producing branches was 300 points. The maximum number of automatically defined functions was 2. The number of arguments for each automatically defined function was 1. The maximum size, H_{adf}, for each of the automatically defined functions, if any, was 300 points. The other parameters for controlling the runs were the default values specified in Koza 1994a (appendix D).

This problem was run on a medium-grained parallel Parsytec computer system consisting of 64 80-MHz Power PC 601 processors arranged in a toroidal mesh with a host PC Pentium type computer. The distributed genetic algorithm was used with a population size of $Q =$ 10,000 at each of the D = 64 demes. On each generation, four boatloads of emigrants, each consisting of B = 2% (the migration rate) of the node's subpopulation (selected on the basis of fitness) were dispatched to each of the four toroidally adjacent processing nodes (Andre and Koza 1996).

4. Results

The best circuit from generation 0 scores 61 hits (out of 72) and achieves a fitness of 3.005 hours and has seven transistors, one diode, and one resistor (not counting the three resistors of the embryo and those embedded in the power supplies).

The best circuit from generation 8 scores 67 hits and achieves a fitness of 2.57 hours and has seven transistors, one diode, and one resistor (not counting the three resistors of the embryo and those embedded in the power supplies).

The best circuit from generation 17 scores 69 hits and achieves a fitness of 2.06 hours while the best circuit of generation 20 scores 72 hits and achieves a fitness of 1.602 hours.

The best-of-run circuit (figure 5) appeared in generation 31 scores 72 hits and achieves a near-optimal fitness of 1.541 hours. This circuit has 10 transistors and four resistors. Its circuit-constructing program tree has 24, 11, and 54 points, respectively, in its three result-producing branches. Its ADF0 has 12 points and is called twice.

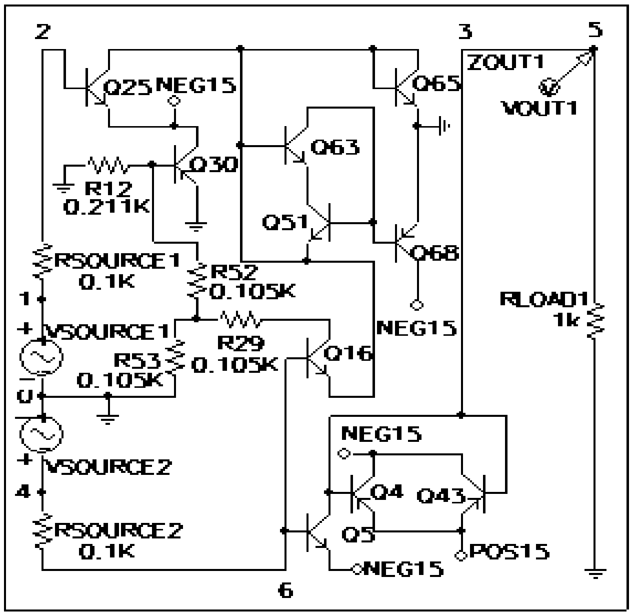

Figure 5 Best-of-run circuit from generation 31.

In comparison, the optimal value of fitness for this problem is known (from Clements 1990) to be 1.518 hours (which is an average of about 21 time steps of about 0.001 hour each for each of the 72 fitness cases).

This genetically evolved result is an electronic circuit that was created without resort to analytical or mathematical methods. Implementation and realization of this evolved result does not require programming of a digital processor or the design of an analog computational circuit. In this instance, the genetically evolved result is the final solution to the original stated problem, namely the problem of designing an electronic device for controlling the flight of an aircraft such that the aircraft flies to an arbitrary destination point in minimal time.

5. Conclusion

The problem of designing an electrical controller to implement the solution to the time-optimal fly-to control problem was used to illustrate that genetic programming can solve problems in a holistic way using the original given ingredients of the problem and without introducing non-indigenous analytical or mathematical techniques.

References

Aaserud, O. and Nielsen, I. Ring. 1995. Trends in current analog design: A panel debate. *Analog Integrated Circuits and Signal Processing*. 7(1) 5-9.

Andre, David and Koza, John R. 1996. Parallel genetic programming: A scalable implementation using the transputer architecture. In Angeline, P. J. and Kinnear, K. E. Jr. (editors). 1996. *Advances in Genetic Programming 2*. Cambridge: MIT Press.

Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming 2*. Cambridge, MA: The MIT Press.

Clements, John C. 1990. Minimum-time turn trajectories to fly-to points. *Optimal Control Applications and Methods*. 11. Pages 39-50.

Gruau, Frederic. 1996. Artificial cellular development in optimization and compilation. In Sanchez, Eduardo and Tomassini, Marco (editors). 1996. *Towards Evolvable Hardware*. Lecture Notes in Computer Science, Volume 1062. Berlin: Springer-Verlag. Pages 48 – 75.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Kinnear, Kenneth E. Jr. (editor). 1994. *Advances in Genetic Programming*. Cambridge, MA: MIT Press.

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

Koza, John R. 1994a. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press.

Koza, John R. 1994b. *Genetic Programming II Videotape: The Next Generation*. Cambridge, MA: MIT Press.

Koza, John R. 1994c. *Architecture-altering operations for evolving the architecture of a multi-part program in genetic programming*. Stanford University Computer Science Department technical report STAN-CS-TR-94-1528. October 21, 1994.

Koza, John R., Andre, David, Bennett III, Forrest H, and Keane, Martin A. 1996. Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996a. Toward evolution of electronic animals using genetic programming. *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*. Cambridge, MA: The MIT Press.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996b. Four problems for which a computer program evolved by genetic programming is competitive with human performance. *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*. IEEE Press. Pages 1–10.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996c. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In Gero, John S. and Sudweeks, Fay (editors). *Artificial Intelligence in Design '96*. Dordrecht: Kluwer. Pages 151-170.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996d. Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming*

1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University. Cambridge, MA: MIT Press.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1997. Evolution using genetic programming of a low-distortion 96 Decibel operational amplifier. *Proceedings of the 1997 ACM Symposium on Applied Computing, San Jose, California, February 28 – March 2, 1997*. New York: Association for Computing Machinery. Pages 207 - 216.

Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: The MIT Press.

Koza, John R., and Rice, James P. 1992. *Genetic Programming: The Movie*. Cambridge, MA: MIT Press.

Quarles, Thomas, Newton, A. R., Pederson, D. O., and Sangiovanni-Vincentelli, A. 1994. *SPICE 3 Version 3F5 User's Manual*. Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA. March 1994.

Version 2 – **G-086** – 6 Pages – CAMERA-READY - Submitted March 25, 1997 to GP-97 Conference