A NEW METHODOLOGY FOR REDUCING BRITTLENESS IN GENETIC PROGRAMMING

Frank W. Moore
Wright State University
Dayton, OH 45435

Dr. Oscar N. Garcia
Wright State University
Dayton, OH 45435

Key Words: Genetic Programming, Brittleness, Fitness Cases, Enhanced Pursuer/Evader Problem

Abstract

Genetic programming systems typically use a fixed training population to optimize programs according to problem-specific fitness criteria. The best-of-run programs evolved by these systems frequently exhibit optimal (or near-optimal) performance in competitive survival environments explicitly represented by the training population. Unfortunately, subsequent performance of these programs is often less than optimal when situations arise that were *not explicitly* anticipated during program evolution. This paper describes a new methodology which promises to reduce the brittleness of best-of-run programs evolved by genetic programming systems. Instead of using a fixed set of fitness cases, the new methodology creates a new set of randomly-generated fitness cases prior to the evaluation of *each generation* of the evolutionary process.

A genetic programming system that evolves optimized maneuvers for an extended two-dimensional pursuer/evader problem was modified for this study. The extended two-dimensional pursuer/evader problem is a competitive zero-sum game in which an evader attempts to escape a faster, more agile pursuer by performing specific combinations of thrusting and turning maneuvers. The pursuer uses the highly effective proportional navigation algorithm to control its trajectory towards the evader. The original genetic programming system used a *fixed* training set of pursuers. Each of these pursuers was uniquely identified by two parameters: the initial distance from pursuer to evader, and the angle that the velocity vector of the evader makes relative to the pursuer/evader line-of-sight at the time the pursuer is launched. The modified system implemented for this project was identical to the original system, except that it used *random* distances and angles to create a new set of fitness cases prior to each generation of the genetic programming run. Best-of-run programs were independently evolved using fixed and randomly-generated fitness cases. These programs were subsequently tested against a large, representative fixed population of pursuers to determine their relative effectiveness. This paper describes the implementation of both the original and modified systems, and summarizes the results of these tests.

Introduction

The two-dimensional pursuer/evader problem (2DPE) is a competitive zero-sum game in which a faster, more agile pursuer is given a limited amount of time to capture an evader as both are traveling across a plane [1]. The game ends favorably for the evader if it manages to stay outside the lethal radius of the pursuer (the maximum distance at which a capture is considered to have occurred) for the duration of the encounter. A solution to the 2DPE problem must incorporate an optimal strategy for maneuvering the evader in a manner that successfully escapes the pursuer, regardless of the initial conditions of the system.

Genetic programming (GP) is an evolutionary programming optimization methodology that shows considerable promise in bypassing bottlenecks of program development and knowledge acquisition encountered when using traditional programming techniques [2]. With GP, *programs* (in the form of parse trees of primitive functions) are automatically created from random combinations of allowable operations and inputs. Reproduction, crossover, and mutation are used to probabilistically breed each generation of programs from the previous generation. With *reproduction*, selected individual programs from one generation are simply copied into the next generation. With *crossover*, two parent programs are selected; a crossover point for each parent program is randomly identified; crossover fragments are created by detaching that portion of the parent

program that lies below the corresponding crossover point in the parent program's parse tree; the crossover fragments of the two parents are exchanged and attached at the crossover point of the new parent; and the resulting programs become part of the next generation. With *mutation*, a random alteration of an individual parent at a single location in its parse tree is performed, and the resulting modified individual is copied into the next generation of programs. Each program is executed against each element of a training population, and the aggregate fitness of each program is estimated by observing overall program performance. Programs demonstrating better performance are more likely to be selected for reproduction, crossover, and mutation than less successful programs. In this manner, the average performance of each generation typically improves. Use of crossover and mutation allows GP to explore new areas of the solution space, reducing the likelihood of premature convergence to local minima. The success of GP depends in large part upon the availability of a *fitness function* that accurately estimates the overall *fitness* of a given program. Although GP cannot guarantee the global optimality of a best-of-run program, GP has produced programs whose performances converge towards near-optimal solution for a wide variety of problems, using a population size which is very small relative to the size of the problem space, within a comparatively small number of generations [2], [3]. The GP methodology represents problem-solving strategies in their most natural form: *programs*.

A Genetic Programming Solution to the Extended Pursuer/Evader Problem

As illustrated in Figure 1, our previous solution to the *extended two-dimensional pursuer/evader problem* (E2DPE) [4] modeled pursuer P and evader E as point masses (rather than massless points) whose motions across a plane were controlled by thrusting forces (applied in the direction of the velocity vector) and turning forces (applied in a direction that is perpendicular to the velocity vector). Both P and E were affected by drag forces and momentum; instantaneous changes in direction were not possible. Since acceleration (a), force (F), and mass (m) are related by the equation a = F/m, the acceleration of P or E depended on its current mass, as well as the magnitude of the applied thrusting and turning forces. The effect of these forces depended on the current state (the position, velocity, and acceleration vectors) of P or E. The maximum distance over which P could pursue E depended on the type of pursuer. P captured E as soon as the distance between them became less than a pursuer-specific lethal radius. The evader performed maneuvers by executing specific combinations of thrusting and turning forces in specific sequences. The optimal strategy for the evader was to maneuver in a manner that maximized the likelihood of successfully evading the pursuer.

Prior to the start of the encounter, the pursuer used the initial state (position, velocity,

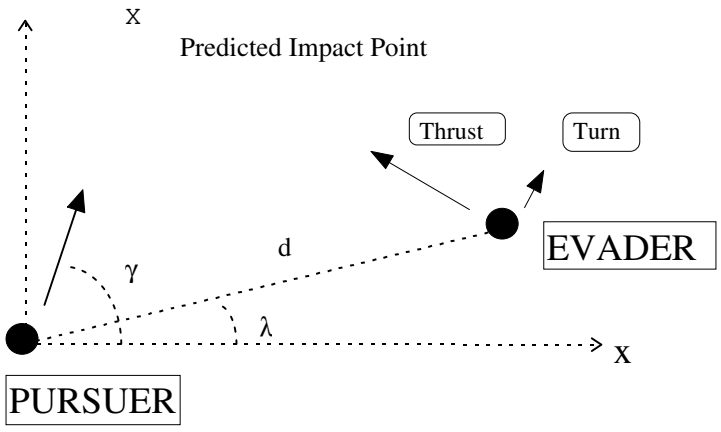

Figure 1. Extended Two-Dimensional Pursuer/Evader Problem.

and acceleration) of the evader to predict an intercept point, and launched itself at maximum thrust towards that point. If the evader failed to maneuver, the pursuer captured the evader at (or very close to) the intercept point. If the evader maneuvered, the pursuer relied on the highly effective *proportional navigation* technique [5] to maintain a constant line-of-sight angle with the evader. Proportional navigation caused the pursuer to accelerate in the direction perpendicular the line-of-sight from the pursuer to the evader; the magnitude of this acceleration was calculated as follows:

$$n_c = N' V_c (d\lambda/dt)$$

where N' is a unitless designer-chosen gain known as the *effective navigation ratio*, and V_c is the pursuer-evader closing velocity vector (the negative rate of change of the distance from the pursuer to the evader). The time derivative of the line-of-sight angle λ is known as the *line-of-sight rate*. The effective navigation ratio for a given guidance system may be determined mathematically from a complex series of computations [6]; for practical guidance systems, optimal values for N' range between 3 and 5 [7].

During each generation of the GP approach, each member of a population of programs for maneuvering the evader was evaluated against each member of a training population of pursuers. Each GP run optimized maneuvers against a single type of pursuer. Proportional navigation was used by all pursuers. The pursuer and evader had complete knowledge of each other's current state (relative position, velocity, and acceleration). A fitness function was used to qualitatively evaluate each evasion program during a simulated encounter; the aggregate fitness of a particular program reflected its fitness when independently executed against each of the pursuers in the training population. The primary measure of program fitness was its ability to evade each pursuer. Fitness-proportionate reproduction, together with crossover, was used to create each new generation. Each run was terminated after a fixed number of generations.

The best-of-run evasion program considered the current state of the evader and pursuer, and output commands that asserted thrusting and turning forces at the appropriate moments in order to accomplish these maneuvers in an optimized manner:

THRUST -- Sets the thrusting force to the specified percentage of the evader's maximum thrust.

TURN – Enacts a turning force equal to the specified percentage of the evader's maximum turning force; the force acts in a direction perpendicular to the evader's current velocity vector; negative values indicate that the turn should be to the left, while positive values cause a turn to the right.

The best-of-run program represented an optimized evasion technique for a specific type of evader and a specific type of pursuer.

The objective of the evader was to survive pursuer attack, regardless of the initial state of the evader and the relative launch position of the pursuer. The initial position of the evader relative to the pursuer at launch time was defined by a fixed set of positions located within the effective engagement space of the pursuer. Note that by a simple rotation of the reference coordinate system (fixed at the launch site of the pursuer), the initial pursuer/evader line-of-sight angle λ could be considered to be the same for all pursuer/evader pairs. For this reason, the following variables were sufficient for describing the initial configuration of each confrontation:

(a) The *line-of-sight distance* between the evader and the pursuer;
(b) The *velocity vector* of the evader at the time the pursuer is launched.

For each pursuer type, the minimum and maximum effective range of the pursuer defined the range of possible initial line-of-sight distances. The line-of-sight distance and the initial velocity vector of the evader were used by the pursuer to predict an intercept point. The pursuer was then launched in the direction of the intercept point, and changed its trajectory only in response to evader maneuvers [8]. The aggregate fitness of a specific program reflected its fitness when evaluated against pursuers launched from each of these initial positions.

A tableau for the E2DPE problem is shown Figure 2. Cartesian coordinates were used to represent components of position, velocity, and acceleration for both the evader and the pursuer. This simple set of terminals proved to be sufficient to allow the GP system to converge to a solution of this problem. This formulation of

the 2DPE fixed the origin of the coordinate system (x = 0, y = 0) at the position of the pursuer. PX and PY thus continually designated the relative displacement from the pursuer to the evader.

Objective:	Determine an optimized evasion strategy for the extended two-dimensional pursuer/evader problem.
Terminal Set:	PX, the displacement in the x-direction from the pursuer to the evader. PY, the displacement in the y-direction from the pursuer to the evader. ***R***, the ephemeral random floating-point constant ranging from -1.0 to 1.0.
Function Set:	ifPX ifPY ifDistance setThrust hardTurn
Fitness Cases:	Numerous fitness cases which differ according to the distance from the pursuer to the evader at the start of the encounter, as well as the acute angle between the initial velocity vector of the evader and the line-of-sight vector from evader to pursuer.
Raw Fitness:	The number of times the distance between the pursuer and evader is less than or equal to the lethal envelope of the pursuer.
Standardized Fitness:	Same as Raw Fitness for this problem.
Hits:	The number of fitness cases that result in capture of the evader prior to the maximum pursuit time of the pursuer. (The encounter is also terminated if the distance between the evader and the pursuer exceeds a pursuer-specific value, at which time the pursuer is considered to have missed the evader.)
Wrapper:	N/A
Parameters:	Population Size M = 100, Maximum Number of Generations G = 21.
Success Predicate:	None.

Figure 2. A Tableau for the Extended Two-Dimensional Pursuer/Evader Problem

For each of these functions, an "argument" consists of any syntactically valid composition of functions, variables, and constants that returns a floating-point value in the range [-1.00 … +1.00]. This normalization of arguments allows arbitrary composition of all functions in the function set. Function ifPX is a two-argument selection function: if the x-displacement of the evader relative to the position of the pursuer is negative, then the first argument is evaluated; otherwise, the second argument is evaluated. Function ifPY is a two-argument selection function defined in a similar manner for the y-displacement of the evader. Function ifDistance is a three-argument selection function: if the current distance between the pursuer and evader is less than the percentage of the pursuer's maximum pursuit distance specified by the absolute value of the first argument, then the second argument is evaluated; otherwise the third argument is evaluated. Each of these functions returns the value of the evaluated argument.

Functions setThrust and hardTurn are single-argument functions. Function setThrust causes the thrust output of the evader to be set to the percentage of its maximum thrust specified by its argument; for example, setThrust (0.9) will set evader thrust to 90% of its maximum possible value. SetThrust ignores the sign of its argument; thrust always acts in the direction of the current evader velocity vector. Function hardTurn causes the evader to execute a turn whose g-force equals the percentage of the maximum allowable turning force of the evader/pilot system specified by its argument; for example, if the maximum turning force is 4 gravities (g's), then hardTurn (0.5) will cause the evader to execute a 2g turn in a direction which is perpendicular and to the right of the current evader velocity vector; the function call hardTurn (-0.5) would result in a 2g evader turn to the left. Both setThrust and hardTurn act

instantaneously, and return the value of their input argument.

In the original GP implementation of the E2DPE problem, fitness cases were identified by two values. The first value, denoted J, was used to identify the initial line-of-sight distance from the pursuer to the evader. If D_{min} and D_{max} denoted the minimum and maximum effective launch distances for the pursuer, then the initial line-of-sight distance D was calculated as follows:

$$D = D_{min} + (J * (D_{max} - D_{min}))$$

The second value, denoted K, identified the angle that the initial velocity vector of the incoming evader makes with the line-of-sight from the evader to the pursuer. Let Θ_v denote this angle. If Θ_{min} and Θ_{max} denote the minimum and maximum value of Θ_v, then Θ_v may be calculated in the following manner:

$$\Theta_v = \Theta_{min} + (K * (\Theta_{max} - \Theta_{min}))$$

D_{min} and D_{max} depend upon the type of pursuer. To maintain the relative geometry illustrated in Figure 1 for the E2DPE problem addressed by this research, Θ_{min} and Θ_{max} described a range of values between 10 and 80 degrees. Each fitness case was denoted by a specific combination of J and K.

The original GP system consistently succeeded in finding a solution that was 100% successful in evading each of the pursuers in the fixed training population. Each run incorporated physical data (including mass) and performance characteristics (such as thrust and drag) of an F-16 evader and an SA-15 pursuer, and assumed an effective navigation ratio N' = 4 for each pursuer. Using a program population size M = 100, the GP system was able to evolve a program implementing an evasion strategy that survived attack from 32 pursuers, each of which was launched from a one of a fixed set of potentially lethal positions. Training using different random number seeds produced best-of-run programs that bore little resemblance to one another; however, each of these best-of-run programs represented an optimized strategy for performing evasive maneuvers that solve the E2DPE problem.

The New Methodology

A frequently-heard criticism of GP solutions is that they are *brittle*. Best-of-run programs frequently exhibit optimal (or near-optimal) performance under conditions explicitly represented by the training population used to evolve the program; however, these same programs often exhibit less-than-optimal performance when situations arise that were *not explicitly* anticipated in the training set. The goal of this paper was to determine whether the brittleness of best-of-run programs could be reduced by creating a new training set of randomly-generated fitness cases prior to each generation of the GP run.

For this purpose, the GP solution to the E2DPE was modified in three ways:

a) The original GP system used a fixed set of distance factors J to specify the initial pursuer/evader line-of-sight distance. The modified system assigned to each fitness case a *randomly-generated J value* between 0.1 and 0.9.
b) The original system also used a fixed set of angle factors K to specify the angle that the initial velocity vector of the incoming evader makes with the line-of-sight from the evader to the pursuer. The modified system assigned to each fitness case a *randomly-generated K value* between 0.1 and 0.9.
c) The set of J and K values used to evolve optimized solutions was fixed throughout each run of the original system. In contrast, new set of random J and K values was generated between *each generation* of each run of the modified GP system.

A finite, uniform distribution of random J and K values was assumed. The actual values for distance and angle were derived from J and K in the manner described above.

Training

To determine whether the modified system produced best-of-run programs that were less brittle than their counterparts evolved using the original system, ten pairs of runs were performed. Each pair of runs allowed the original and modified systems to independently generate best-of-run programs. The original system used a fixed set of fitness cases, while the modified system randomly generated a new set of fitness cases prior to evaluating each generation of programs; all other conditions were identical for each pair of runs.

The training population for Training Runs 1-5 consisted of 16 fitness cases; the fixed training population for each of these runs was described as follows:

J ∈ {0.3, 0.45, 0.55, 0.7};
K ∈ {2/9, 3/9, 4/9, 5/9}, identifying angles of 20-50 degrees.

In contrast, each of the 16 fitness cases in each training set used by the modified system was identified by a specific combination of randomly-generated values for J and K, as described above.

Training Runs 6-10 used 32 fitness cases; the fixed population was described as follows:

J ∈ {0.45, 0.5, 0.55, 0.6};
K ∈ {1/9, 2/9, 3/9, 4/9, 5/9, 6/9, 7/9, 8/9}, identifying to angles of 10-80 degrees.

The 32 fitness cases in the training sets used by the modified system were identified by randomly-generated J and K values.

Training Runs 1-10 evolved ten pairs of best-of-run programs. Each pair consisted of one program evolved using a fixed training set, and another program evolved under conditions where a new randomly-generated training set was created prior to each generation of the GP run.

Test Results

After best-of-run programs were independently identified by each training run, each of these programs was subsequently tested against a large, representative fixed test population of 128 pursuers. The results of these tests are tabulated in Figure 3.

For fixed training sets, Tests 1-5 demonstrate that using a few well-chosen representative fitness cases is more important than using a larger number of fitness cases. Programs evolved in Training Runs 1-5 used a wider range of distance factors, and focused efforts only on training cases where the initial angle between the evader velocity vector and line-of-sight was relatively small. Use of random fitness cases did not help evolve less brittle best-of-run programs in these Tests 1-5; aggregate performance of programs evolved using either fixed or random fitness cases in Tests 1-5 differed by less than 1%.

In spite of the fact that the training population for Training Runs 6-10 was double the size of that used by Training Runs 1-5, the best-of-run programs evolved using fixed fitness cases proved in general to be more brittle. We attribute this result to overtraining with a less representative fixed training population. For Tests 6-10, the aggregate performance of programs evolved using randomly-generated training sets was nearly 9% better than those evolved with fixed training sets. Thus, the new training methodology helped evolve programs that in general were significantly less brittle than their fixed-fitness-case counterparts.

	FITNESS CASES	
TEST	Fixed	Random
1	110 (85.9%)	105 (82.0%)
2	109 (85.2%)	111 (86.7%)
3	106 (82.8%)	108 (84.4%)
4	100 (78.1%)	103 (80.5%)
5	111 (86.7%)	103 (80.5%)
6	114 (89.1%)	105 (82.0%)
7	90 (70.3%)	112 (87.5%)
8	106 (82.8%)	106 (82.8%)
9	87 (68.0%)	114 (89.1%)
10	110 (85.9%)	114 (89.1%)

Figure 3. Test Results

Large-scale program testing of optimized solutions to the E2DPE problem has consistently demonstrated that the most difficult encounters for the evader take place when the both J and K are small, i.e., whenever the pursuer is launched from close range and the angle between the initial velocity vector of the evader and the line-of-sight is small. 25% of the fixed training population used during Training Runs 1-5 were launched from a relatively short distance (D = 0.3), and 50% used relatively small angles (20 and 30 degrees). Because no close-range distances were included in the fixed training population of Training Runs 6-10, the best-of-run programs evolved during these runs were incapable of surviving these more difficult pursuer/evader encounters during subsequent testing against a large, representative population of pursuers. In contrast, the random distances used during Training Runs 6-10 were likely to include at least a few short-range, small-angle launches in each training population; as a result, the best-of-run programs were likely to be more capable of handling a much higher percentage of

the most difficult encounters during subsequent testing.

Conclusions

The ideal training population used to evolve and optimized program via GP would be a perfect representation of all situations which might be encountered by the best-of-run program during subsequent use. Practical considerations limit the size of the training population, thus reducing the percentage of situations explicitly anticipated by that population. Best-of-run programs evolved by GP systems frequently exhibit an unacceptably high amount of brittleness, particularly when the training population fails to adequate represent difficult situations that may occur. The results of this study suggest that the brittleness of best-of-run programs evolved by GP applications can be reduced by creating a new training population of randomly-generated fitness cases prior to the evaluation of each generation of programs. This new methodology is particularly beneficial when the fixed training population does not adequately represent the full range of difficult situations that may arise during subsequent program testing. Our results encourage the application of this methodology to other complex strategy optimization problems.

References

[1] Hamalainen, R. P. and H. K. Ehtamo (eds.), 1990. Differential Games -- Developments in Modeling and Computation, Lecture Notes in Control and Information Sciences Vol. 156, Springer-Verlag.

[2] Koza, J. R., 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press.

[3] Kinnear, K. E. Jr. (ed.), 1994. Advances in Genetic Programming, MIT Press.

[4] Moore, F. W. and O. N. Garcia, 1997. "A Genetic Programming Approach to Strategy Optimization in the Extended Two-Dimensional Pursuer/Evader Problem", in Koza, J. *et al*, Proceedings of the Genetic Programming 1997 Conference, Palo Alto, CA.

[5] Ball, R. E., 1985. The Fundamentals of Aircraft Combat Survivability Analysis and Design, AIAA Education Series, AIAA Inc.

[6] Barron, R. L., 1995. "Reduced-Computation End-Game Steering Laws for Predictive Guidance", in Journal of Guidance, Control, and Dynamics, Vol. 18 No 2, March-April 1995, pp. 306-315, American Institute of Aeronautics and Astronautics, Inc.

[7] Ramo, S. and A. Pucket, 1959. Guided Missile Engineering, McGraw-Hill, pp. 176-180.

[8] Zarchan, P., 1990. Tactical and Strategic Missile Guidance, Progress in Astronautics and Aeronautics Vol. 124, American Institute of Aeronautics and Astronautics.

Shinar, J. and D. Steinberg, 1977. "Analysis of Optimal Evasive Maneuvers Based on a Linearized Two-Dimensional Model", in Journal of Aircraft, Vol. 14, August 1977, pp. 795-802.

Author Biographies

A Dayton Area Graduate Studies Institute scholarship recipient, **Frank Moore** is currently completing his doctorate degree in Computer Science & Engineering at Wright State University. His dissertation advisor, **Dr. Oscar Garcia,** is NCR Distinguished Professor and Chair of the Department of Computer Science & Engineering at Wright State.