Parametric Regression Through Genetic Programming

Edwin Roger Banks, James Hayes, and Edwin Nunez

COLSA Corporation, 6726 Odyssey Drive, Huntsville, AL 35806
{rbanks, jhayes, enunez}@colsa.com

Abstract. Parametric regression in genetic programming can substantially speed up the search for solutions. Paradoxically, the same technique has difficulty finding a true optimum solution. The parametric formulation of a problem results in a fitness landscape that looks like an inverted brush with many bristles of almost equal length (individuals of high fitness), but with only one bristle that is very slightly longer than the rest, the optimum solution. As such it is easy to find very good, even outstanding solutions, but very difficult to locate the optimum solution. In this paper parametric regression is applied to a minimum-time-to-target problem. The solution is equivalent to the classical brachistochrone. Two formulations were tried: a parametric regression and the classical symbolic regression formulation. The parametric approach was superior without exception. We speculate the parametric approach is more generally applicable to other problems and suggest areas for more research.

1 Introduction

Mathematicians sometimes use parametric formulations of expressions. Rather than write y(x), for example, they would write y(t) and x(t) where *t* is an arbitrary parameter that generates the corresponding values of y and x. In this paper we explore use of the parametric formulation in genetic programming[1] for a minimum-time-to-target problem and show it to be superior.

2 Minimum-Time-to-Target Problem

We considered a problem where a missile was launched at a moving target. Our goal was to find the fastest path from launch site to this target. For simplicity, the problem was initially posed with a stationary target. In this paper we will only report on solutions for this simple situation since we were attempting to gain in-depth knowledge of a solution methodology. For the GECCO 2004 conference we expect to add various complicating factors like thrust, drag, cross winds, movement of the target (possibly evasive) and allowance for proximity to target. A first simplified task was to use genetic programming (GP) to rediscover the well-known *brachistochrone* solution. The

[1] See [1] for an excellent tutorial introduction to genetic programming.

missile interception problem (Figure 1a) is analogous to the brachistochrone problem if an axis inversion is performed (Figure 1b). The brachistochrone is simply the path of least time that a bead on a frictionless wire would take to fall under the influence of gravity to reach a fixed point. It is minimum when the wire is shaped like a cycloid with the y-axis downward.

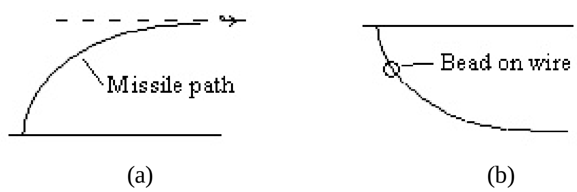

Fig. 1. (a) Missile launch to a target (b) Analogy to bead on wire (brachistochrone).

3 The Brachistochrone Problem

The solution to the brachistochrone curve is well known [2,3]. Isaac Newton originally solved it in about a day. It is expressed parametrically as:

$$X = a(\theta - \sin(\theta)) \tag{1}$$

$$Y = a(1 - \cos(\theta)) \tag{2}$$

In these equations X and Y are the coordinates representing bead position (Figure 1b), and θ is a parameter. The parametric form is used because there does not exist a closed-form solution in terms[2] of Y = Y(X).

3.1 The GP Formulation of the Brachistochrone

For our chromosome we chose two genes[3], representing the expressions for X and Y, respectively. The set of operators used were +, -, *, /, ^, S, and C, where ^ represents exponentiation to a possibly non-integer power, S is the sine, and C the cosine function. For terminal symbols, we pre-defined immutable constants 0, 1, and p (for pi) and also allowed an arbitrary mutable constant, k. Multiple occurrences of k in a chromosome can take on different values and are mutated independently. However, during recombination (crossover), k preserves its value in the offspring.

[2] It is interesting to note that there does indeed exist the inverse solution, X = X(Y) in terms of square root and trigonometric functions. The X(Y) form is, however, multi-valued.

[3] Terminology: a chromosome is made up of several genes, each of which in turn is made up of operator and terminal symbols. The entire chromosome is packed as a linear string.

We also chose to use a Karva-like notation [4, 5, 6] as the chromosome representation type[4]. (In brief, if you draw the tree for a symbolic expression, then scan it in left-to-right, top-to-bottom order, the symbols form a Karva expression. Unlike Karva, our notation allows operators throughout the entire gene.) We also made a few runs with RPN as a comparison. We chose a gene length of 21 symbols.

3.2 Fitness Evaluation and the Solution Landscape

The most interesting findings related to this problem were associated with difficulties encountered during fitness evaluation. The most important is described in 3.3 below.

To assess fitness, we performed a numerical integration by successively evaluating the genes for X and Y at increasing values of θ. Over each interval of X_i, Y_i, to X_{i+1}, Y_{i+1}, we simply computed the time increment and added it to the total time. This is the time that has to be minimized. In the limit, as the step size for $\Delta\theta$ approaches zero, the true brachistochrone time is approached.

3.3 Brush-like Solution Space

Despite much tuning of our GP parameters (such as mutation and recombination rates), we did not discover the exact parametric solution. But we discovered many "solutions" that were very good and came quite close to the minimum time dictated by the brachistochrone. When plotted, the best discovered solutions lie on top of the true brachistochrone curve except for a few pixels. Figure 2 below shows some intermediate solutions to the intercept problem for different endpoints (not to scale).

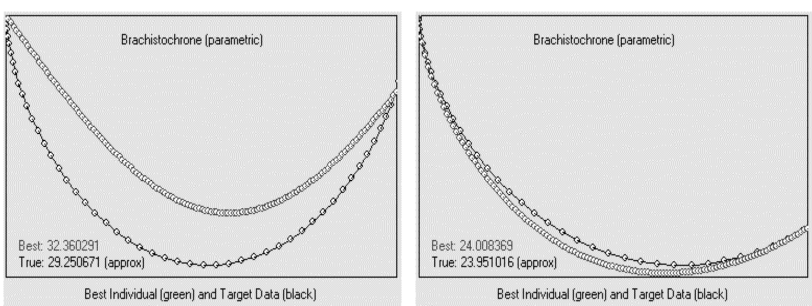

Fig. 2. Comparisons of parametric and non-parametric solutions

We may have determined why the genetic programming software was unable to discover the true brachistochrone. This problem needs additional research to determine if

[4] Our research has shown that Karva appears to evolve to a solution a bit faster than our other representation types (e.g., RPN or Reverse Polish Notation) although it runs at a *slower* generation rate. A desired fitness is reached sooner in time with Karva, while RPN runs more generations per second.

the reason we give below is the one causing the GP system to fail in its attempt to reach the true brachistochrone.

Consider that irrespective of any monotonic expression the GP discovers for X, it can usually find a corresponding Y expression to approach the desired curve very closely. Likewise, for any monotonic Y expression, it can also find an expression for X. This leads us to the conclusion that the resulting fitness landscape is very much like an inverted scrub brush where each bristle is a local minimum peak. There are many bristles, just as there are many monotonic expressions for X. As figure 3 illustrates, only one of the bristles is in the form of the exact best solution and it is almost indistinguishable from the other bristles in terms of its peak value[5].

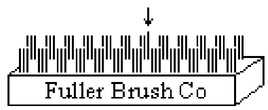

Fig. 3. Brush-like structure of fitness landscape.

Thus, it would have been quite fortuitous to discover the exact parametric brachistochrone by accident. Nevertheless, we discovered several near misses, one of which included the exact solution embedded in a larger solution with other symbols.

This led us to study whether it was possible that some of the other solutions found by the GP system may have been exact as well. It is very difficult, however, to recognize such solutions if they do appear. For example, since θ is an arbitrary parameter, any substitutions, such as $t = \tan(\theta)$ are perfectly legitimate. Such substitutions change the visual appearance of the equation greatly so that it might not be recognized as Newton's result. We considered, but have not yet implemented, the idea of *pseudo-equivalence testing* by supplying a small number of random values to the discovered expressions of X and Y and likewise to the true brachistochrone. If the results are identical, then the expressions are probably identical, with higher and higher probability depending on the number of tests applied. It should be noted that one must take into account that the intrinsic system runtime libraries are typically not maximally accurate to the last decimal place when making the comparisons.

Nevertheless, given our scrub brush analogy, we believe that our GP *never found the true parametric solution*, even after many hours of running.

Just as the scrub brush has many very good bristles, a formulation of a GP problem in terms of parametric equations should be able to do symbolic regressions *much* faster. This result also deserves investigation since it may apply more generally to other

[5] Not to be misleading, there are also many inferior individuals (shorter bristles) in the full fitness landscape. The point is that there are a great many exceptionally good individuals, and it is these that form the scrub brush analogy.

types of GP problems[6]. The parameter (θ) is somewhat like a *slack* variable that gives GP more search freedom, adding many good peaks to the fitness landscape.

On the other hand, the brachistochrone has a peculiarity in that the initial slope, dy/dx, is infinite and thus any non-parametric solution is constrained by this fact. Since the total time is heavily biased toward the initial part while the bead on the wire is picking up speed, it may simply be that our choice of problem area is the full explanation for our favorable parametric results. If so, the applicability of this technique will be narrower.

4 An Additional Formulation of the Brachistochrone Problem

Thus far we have attempted to solve for the expression with minimum time to the target X_{target}, Y_{target} for various, but fixed values of X_{target}, Y_{target}. It is also possible with our GP system to evolve a more general solution of all brachistochrones in terms of multiple independent variables. This means we wish to evolve expressions for X and Y in terms of X_{target}, Y_{target}, and θ. Thus, in equations (1) and (2) above, "a" becomes a function of X_{target}, Y_{target}. To explore these solutions we created a training data set of 50 known brachistochrone times as a function of X_{target}, Y_{target} choosing the target values randomly[7]. Then, adding two variables, x, and y, to our chromosome terminal set, evolved generic solutions. This was done both parametrically and non-parametrically. The results (below) were not nearly as fit as evolving a particular brachistochrone to a particular X_{target}, Y_{target}. Nevertheless, the parametric formulation remained superior.

5 Comparison of Parametric and Non-Parametric Solutions

We performed many comparisons of the parametric and non-parametric GP solutions. Without exception, the parametric version was superior, despite a variety of values for the GP parameters (recombination rates, population sizes, two types of mutation rate, and about a dozen other controllable parameters). Within groups of 10 runs, each typically for a half-hour or longer, some particular non-parametric runs were better than some parametric runs, but the best-of-run was always superior for the parametric

---

[6] Just before submission of this paper we constructed and ran an additional problem, namely a classical symbolic regression in an attempt to find the non-parametric expression Y(x) and the parametric expressions $X(\theta)$, $Y(\theta)$ that best matched a table of 16 numbers. Two sets of 10 runs were made for each formulation, *and in this case we did **not** find the parametric solution to be superior*. However the particular formulation may not have been favorable and we will continue to investigate further. It may be that the decision to use parametric or non-parametric formulation is strongly dependent upon the problem. Nevertheless, the parametric approach presented herein should be considered for your problem. We believe the parametric formulation will continue to be the favorable formulation for more complex minimum-time-to-target problems. Additional results will be presented at the GECCO conference.

[7] We also included 2 additional known solutions not chosen randomly for 52 total test cases.

runs and the best results were often near-perfect. The parametric formulation was clearly superior in all cases as shown in the Table 1 below.

Values in the result table below are the percent error in time as compared to the true minimum time. The true minimum was computed two different ways that differed very slightly (e.g., 23.951 and 23.971) and the average was used as the reference value for comparison. The downward force of gravity was assumed to have the value 0.5. Run results are averaged over 10 or more smaller sub-runs (footnoted if less than 30 minutes). The best-of-run and average-of-run times are shown. Run #7 was the best run, each sub-run being 4 hours in length. Run #14 was the best run for which we had a comparison to the non-parametric formulation. Some runs were deliberately crippled, providing fewer operators to work with or varying GP parameters such as population size, etc.

Note that there are *no exceptions* to the superiority of the parametric formulation in terms of the best-of-run times, and only 1 exception (run #12) where the average-of-run was very slightly better for the non-parametric formulation. This exception was probably not statistically valid because it was among the shortest of the runs, with only 10 minutes for each sub-run.

Table 1. Comparison of parametric and non-parametric runs

Run #	Best / Average % error		Footnotes
	Parametric Y(t), X(t)	**Non-parametric Y(x)**	
1	0.35 avg	0.66 avg	1, 4
2	0.16 avg	0.52 avg	1, 4
3	0.159 / 0.674	1.087 / 1.978	1, 4, 7, 8
4	0.070 / 0.364	0.941 / 1.301	1, 4, 7
5	0.067 / 0.163	0.104 / 0.522	1, 4
6	0.381 / 0.722	0.620 / 1.03	2, 5
7	0.056 / 0.092	N.A.	1, 4,11
8	0.135 / 0.441	0.543 / 1.166	1, 4
9	0.187 / 0.679	0.823 / 1.108	1, 4
10	0.717 / 1.810	1.470 / 2.055	1, 4
11	0.124 / 0.503	N.A.	1, 4,11
12	0.534 / 1.570	0.927 / 1.51	1, 4, 9
13	0.067 / 0.164	0.106 / 0.522	1, 4
14	0.0565 / 0.151	0.0760 / 0.232	3, 6
15	34.47 / 36.08	35.64 / 37.18	10
16	34.57 / 36.26	36.30 / 38.27	10

Table Footnotes:

1. End-point (target X, Y) is 86.26193, 31.61920.
2. End-point is 100, 10.

3. End-point is 52.48684, 30.68447.
4. Reference time (true brachistochrone time) is 23.961.
5. Reference time is 29.358,
6. Reference time is 18.175.
7. Sub-runs were only 5 minutes each for runs 3 and 4.
8. Reverse Polish Notation used for representation time.
9. Sub-runs were only 10 minutes each for run 12.
10. Generic brachistochrone runs (see "Additional Formulation…" above). Fitness values for generic runs are in percent over the total of true times for 52 mostly randomly selected end-points. Best-of-run and average-of-run are reported.
11. Non-parametric runs were not made.

One of the better solutions found was:

$$X: \quad ((\theta + \sin(26.0075373) + \theta) * \theta) + \theta \tag{3}$$

$$Y: \quad \theta * (11.2222191 - (((\cos(\theta) * 0.333373901) * \ln(\theta)) + \theta)) . \tag{4}$$

where the trigonometric functions expect arguments in radians.

Varying parameters such as recombination rates, representation type, population size, operator choice, etc., within reason all had a smaller effect on fitness than did choice to use the parametric representation.

To summarize, it appears that if you seek to discover an exact solution, then you might avoid a parametric formulation of the problem[8]. However if you merely seek an outstanding solution that is likely to be very close but <u>not</u> exact, then embrace parametric formulation. We intend to conduct more research on GP and parametric regression to establish its capabilities.

6 More Complex Minimum-Time-To-Intercept Problems

As indicated earlier, we are planning to add thrust, drag, crosswind, target movement and other complications to the model. The result should be available during the GECCO conference.

[8] Of course, for the brachistochrone, one is forced to seek a parametric solution since no closed-form solution exists.

7 Comparison to Prior Work

Reference [7] describes a piece-wise linear approximation to the brachistochrone discovered by a genetic algorithm.

Reference [8] is an example of using GP to intercept a moving target with evasive maneuvers, and attempts to be a high-fidelity model, using simulation code based on actual missile parameters. The cost of this fidelity was that very few generations could be run. Nevertheless GP found improved solutions.

8 Final Remarks on GP and Parametric Solutions

Anybody considering the use of GP for the search of parametric solutions should consider our experience, as indicated below. It may help them avoid some pitfalls and guide their research by what we found.

Pitfall. Whatever $\Delta\theta$ step we used for the numerical integration, the GP system would sometimes evolve a solution that multiplied $\Delta\theta$ by a large constant to make it no longer small. We had assumed that we could ignore the last interval near the target since the integration step was *small*. Wrong!

Rule #1: GP will evolve to take advantage of any assumptions you make in formulating the problem. It can (and did) discover better-than-physically possible "solutions" until we corrected our formulation. If your run-time library makes assumptions or is inaccurate, it will also attempt to take advantage of it, too.

Rule #2: GP will evolve a solution that makes you satisfied with the result, whether or not it is a good solution or has any relationship to theory or practice. **The evolution environment for fitness evaluation always includes the researcher!**

References

1. Koza, John R., Introduction to Genetic Programming, Tutorial, GECCO-2003, Chicago (2003)

2. Hiebert, K.L.: ACM Trans. Math. Software **8 (**1982) 5-20

3. Hildebrand, Francis B.: Methods of Applied Mathematics. 2nd edn. Prentice-Hall (1965)

4. Ferreira, C., 2001. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. Complex Systems, 13 (2): 87-129

5. Ferreira, C.: Gene Expression Programming in Problem Solving, WSC6 Tutorial, (2001)

6. Ferreira, C., Gene Expression Programming in Problem Solving. In R. Roy, M. Köppen, S. Ovaska, T. Furuhashi, and F. Hoffmann, eds., Soft Computing and Industry – Recent Applications, pages 635-654, Springer-Verlag, 2002

7. Erickson, D., Killmer, C., Lechner, A: Solving the Brachistochrone Problem with Genetic Programming, 860-864

8. Moore, F.W.: A Methodology for Missile Countermeasures Optimization under Uncertainty. Evolutionary Computation **10** (2002) 129-149

9. Eiben, A.E., and Smith, J.E.: Introduction to Evolutionary Computing, Springer, (2003)