SMCGP2: Finding Algorithms that Approximate Numerical Constants Using Quaternions and Complex Numbers

Simon Harding
Department Of Computer Science
Memorial University
Newfoundland, Canada
simonh@cs.mun.ca

Julian F. Miller
Department of Electronics
University of York
York, UK
jfm7@ohm.york.ac.uk

Wolfgang Banzhaf
Department Of Computer Science
Memorial University
Newfoundland, Canada
banzhaf@cs.mun.ca

ABSTRACT

Self Modifying Cartesian Genetic Programming 2 (SMCGP2) is a general purpose, graph-based, developmental form of Cartesian Genetic Programming. Using a combination of computational functions and special functions that can modify the phenotype at runtime, it has been employed to find general solutions to a number of computational problems. Here, we apply the new SMCGP technique to find mathematical relationships between well known mathematical constants (i.e. pi, e, phi, omega etc) using a variety of functions sets. Some of formulae obtained are distinctly unusual and may be unknown in mathematics.

Categories and Subject Descriptors

I.2.2 [**ARTIFICIAL INTELLIGENCE**]: Automatic Programming; D.1.2 [**Software**]: Automatic Programming

General Terms

Algorithms

Keywords

Genetic programming, developmental systems

1. INTRODUCTION

Self-Modifying Cartesian Genetic Programming (SMCGP) [1] is a form of Genetic Programming (GP) that includes time (i.e. iteration) in the genotype-phenotype map. Through the inclusion of self-modifying (SM) functions in its function set, SMCGP allows a genotype to be iterated into a possibly unlimited sequence of phenotypes. Each phenotype encodes a computational function that can be assessed for fitness. It has been used to find programs that can approximate fundamental mathematical constants, π and e to arbitrary precision [2]. The method showed that it was possible to discover mathematically exact new algorithms (for π) and re-discover others (for e) that are based on the recurrent and iterative methods used by mathematicians. SMCGP was able to find many such programs with ease, and some of them were simple enough to be interpretable by humans. This paper applies a new form of SMCGP, called SMCGP2, presented in a companion paper [3] which is a two-dimensional extension of SMCGP. We apply SMCGP2 to the problem of approximating an extended suite of well known mathematical and physical constants (see Section 3). The main novelty of this paper is that we have further extended the representation here so that SMCGP2 can handle complex and quaternion type data.

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

2. SMCGP2

In summary, the SMCGP2 genotype is a grid of nodes, where each node encodes several elements. Each node contains genes for its function (either computational or self modifying), a list of which other nodes it connects to, a numeric constant and parameters used by the SM functions. During execution, a copy of the genotype is made, which becomes the phenotype. This phenotype program is iterated, and between iterations the phenotype can be altered by SM functions used in the previous iteration. See [3] for a complete description. SM functions include: the ability to duplicate sections of the phenotype, and insert them elsewhere in the phenotype, copying sections and overwriting existing nodes, deleting sections, rows or columns, and adding blank rows or columns. Computational nodes, in this instance, are conventional mathematical operations. Any addition/removal of nodes in SMCGP2 results in a valid phenotype, as nodes are connected to each other using 'relative addressing'. SMCGP2 uses the same input/output strategy as SMCGP, where special functions are used to obtain the value of the 'next available' input. An 'output' function indicates which nodes can be used for outputs.

SMCGP2 uses a simple, 1+4 evolutionary strategy is applied. Genotypes are 5 nodes high, by 20 nodes wide. A mutation rate of 5% is used, with each gene having an equal probablity of being changed. The maximum SM size is 1,000 nodes. The maximum size of a phenotype is 100,000 nodes. The maximum number of SM operations that can be applied per iteration set to two. Phenotypes are parsed backwards (i.e. bottom-right to top-left).

The fitness of an individual is the difference between the target value and the program's output on the final iteration. Up to 20 iterations are allowed, but if there is convergence the fitness function will stop iterating. For each iteration, the program is given the iteration index, the previous output and the list of other constants as inputs. On the first iteration, the value passed as the previous output is 0. This fitness function is similar to that used in previous work [2], but does not penalize the evolved program if it does not use the developmental properties of SMCGP.

Task		Computational Effort		
Target	Inputs	Real	Complex	Quaternions
Function Set A				
π	E, φ, Ω	25,837,868	113,737,960	58,508,652
E	π, φ, Ω	9,801,632	2,569,164	40,940,964
Ω	E, φ, π	31,340,260	9,671,436	36,978,508
φ	E, π, Ω	5,790,404	11,766,552	2,276,604
α	E, φ, π, Ω	63,629,460	23,667,012	70,381,488
Function Set B				
π	E, φ, Ω	63,797,488	53,984	28,226,224
E	π, φ, Ω	5,246,040	1,347,216	25,997,720
Ω	E, φ, π	17,834,924	28,696,284	32,902,888
φ	E, π, Ω	3,393,576	11,721,056	2,425,424
α	E, φ, π, Ω	31,803,592	8,970,248	4,571,280
Function Set C				
π	E, φ, Ω	53,428	56,500	20,506,088
E	π, φ, Ω	1,193,832	6,808	43,645,400
Ω	E, φ, π	31,900,424	9,951,028	28,677,960
φ	E, π, Ω	7,448,140	14,147,208	2,960,640
α	E, φ, π, Ω	17,460,480	15,733,928	7,485,728

Table 1: Computational Effort for each task.

Function Set	Real	Complex	Quaternion
A	53.6	49.6	24
B	56	83.2	71.2
C	84.8	86.4	69.6

Table 2: Average percentage success per function set for each number system.

3. RESULTS

In our previous work, experimentation was limited to just π and e. Here the number of target constants is increased to cover a range of interesting constants. Additionally, different conditions were tested (i.e. different input values and function sets) Each task was tested with all three number types.

π and e are familiar mathematical constants. A large number of known ways exist to approximate these numbers. φ is the Golden Ratio (≈ 1.61803). Known solutions include continuous fractions, an irrational quadratic and an infinite series. The omega constant, Ω, is ≈ 0.567143.The Fine Structure Constant α (≈ 0.007297) is a fundamental physical constant that relates several other physical constants. The value is determined by measurement, however several numerological approximations also exist.

Three different function sets were used: A,B and C. Due to space, we have omitted the complete listing. However, function set A contains primitive functions, B adds the ability to use evolved numerical constants and set C add trigonometric and power functions. Each group ,contains the previous groups functions i.e. group B also contains the functions in group A, and C contains those in both A and B.

SMCGP2 was able to find many solutions to the problems given. Tables 1 and 2 shows that in general, the more complete the function set, the more likely solutions are to be found. However, a large majority of the solutions are very involved, especially when SMCGP2 uses development to produce iterative or recursive programs. It appears that, in general, it is slightly easier to evolve solutions using complex numbers than either quaternions or real numbers, and that using quaternions results in the worst average performance. However, it is difficult to compare these results between number systems as the function sets are not the same. We will release extensive appendices can be found at `http://www.cs.mun.ca/~simonh/smcgp2/constants/`. The following examples are illustrative of the results.

The following equation appears to link e to Ω, using complex number operations, although it never uses the imaginary components: $e = 0 + inv(\Omega)^{inv(\Omega)}$

In another example using complex numbers, we find this curious relationship between Pi, e, some integer constants and phi:

$$\varphi = cAbs(\sqrt{(}coth(3) + \tan(coth(3)^{\pi})) - exp(\pi + \pi + \pi + \pi) + \frac{\pi + \pi + \pi + \pi}{\pi}^{\pi+\pi} \times \tan(\pi + \pi + \pi + \pi) \times \frac{\pi + \pi + \pi + \pi}{\pi} \times 2024 - \tanh(\cos(\tanh(e))) + exp(\pi + \pi + \pi + \pi))$$

Here, the imaginary components do seem important to the calculation. Whilst this equation looks like it may simplify, we discovered that it only works because of the implementation. Minor rounding errors, typically only visible on the last digit of the double precision numbers, become compounded and it appears there is some inflection point in the calculation. For example $\tan(\pi + \pi + \pi + \pi)$ does not resolve to 0, as the summing of four π values does not produce exactly 4π.

4. CONCLUSIONS

The results show that quaternions are equal, or better, in terms of success 7 times out of 15 experiments. Similarly, complex numbers are equal or better than reals for 10 out of 15 experiments. We can therefore say that the number systems and their function sets do seem preferable under some, but not all, conditions.

We are convinced that serious efforts in using evolutionary algorithms, and Genetic Programming in particular, with more complex number systems will lead to surprising results in the future.

5. ACKNOWLEDGMENTS

WB acknowledges funding from Atlantic Canada's HPC network ACENET and by NSERC under the Discovery Grant Program RGPIN 283304-07.

6. REFERENCES

[1] S. Harding, J. F. Miller, and W. Banzhaf. Self-modifying cartesian genetic programming. In H. Lipson, editor, *GECCO*, pages 1021–1028. ACM, 2007.

[2] S. Harding, J. F. Miller, and W. Banzhaf. Self modifying cartesian genetic programming: finding algorithms that calculate pi and e to arbitrary precision. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 579–586, New York, NY, USA, 2010. ACM.

[3] S. Harding, J. F. Miller, and W. Banzhaf. SMCGP2: Self modifying cartesian genetic programming in two dimensions. In *Accepted for publication in GECCO 2011*, GECCO '11, New York, NY, USA, 2011. ACM.