Function Optimization using Cartesian Genetic Programming

Julian F. Miller
Electronics Department
University of York, UK
julian.miller@york.ac.uk

Maktuba Mohid
Electronics Department
University of York, UK
mm1159@york.ac.uk

ABSTRACT

In function optimization one tries to find a vector of real numbers that optimizes a complex multi-modal fitness function. Although evolutionary algorithms have been used extensively to solve such problems, genetic programming has not. In this paper, we show how Cartesian Genetic Programming can be readily applied to such problems. The technique can successfully find many optima in a standard suite of benchmark functions. The work opens up new avenues of research in the application of genetic programming and also offers an extensive set of highly developed benchmarks that could be used to compare the effectiveness of different GP methodologies.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Genetic programming, function optimization

1. INTRODUCTION

Function optimization benchmarks have become highly developed particularly through criteria for competitions at conferences [2]. Almost without exception, optimization techniques operate in the domain of the objective function so that the algorithms manipulate the arguments of the optimization function. We refer to this as a *direct* representation. It is not clear, *a priori* that evolving solution using a direct representation will allow optimization algorithms the best chance to optimize these complex functions. However, to date except for a couple of studies indirect approaches to function optimization have almost totally been neglected. Pujol and Poli proposed a parameter mapping approach (PMA) in which genetic programming is used to evolve a program which from a collection of random constants and a *single* independent variable of the optimization function (they call it a parameter) returns an improved value for this variable [5]. The *same* evolved program is used to produce improved values for all the independent variables. Walker and Miller used CGP for evolving programs that wrote binary bits on a finite tape, thus outputing binary strings [7]. They showed that this was an effective method of solving two binary problems. They applied a similar technique to three real-valued optimization problems [8].

Here we propose a new approach using Cartesian Genetic Programming [4] to construct a program that maps numeric constants to independent variables. Apart from possible advantages this approach might bring to the optimization of difficult functions, the proposed approach offers the genetic programming (GP) community a highly developed set of benchmarks that could be used to compare the effectiveness of various GP approaches [3].

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

2. RELATIONS BETWEEN COORDINATES OF OPTIMA

In general if one was trying to optimize a complex machine involving d variables, one would expect many of the variables to be mathematically related. Indeed, imagine that the optimization function is differentiable. The normal procedure for obtaining the optima would be differentiate the function with respect to each variable and equate the partial derivatives to zero (actually optima may lie on boundaries, we don't consider this situation here). This would give d equations, in many cases these would relate the coordinates of the optima to each other. Here is a simple example.

2.1 An illustrative function

Consider the non-separable, non-convex optimization function [1] given by:

$$f(x) = \sum_{k=1}^{d} [(k+1)\sin(x_1) - x_{k+1}]^2 \quad (1)$$

Taking partial derivatives we obtain:

$$\frac{\partial f}{\partial x_1} = 2\cos(x_1)\sum_{k=1}^{d}(k+1)[(k+1)\sin(x_1) - x_{k+1}] \quad (2)$$

$$\frac{\partial f}{\partial x_k} = -2[(k+1)\sin(x_1) - x_{k+1}] \quad (3)$$

$$(4)$$

Thus optima occur when $(k \geq 2)$

$$x_{k+1} = (k+1)\sin(x_1) \quad (5)$$

So the set of optima are a function of a *single* variable x_1! This implies that if a method could be found that could relate x_1 and x_k to each other through a mathematical expression, it ought to be able to find the coordinates of the optima more easily than by random variation of the variables x_k. Classic direct function optimization techniques attempt to discover numerical values for each variable independently, whereas genetic programming can discover functional relationships between variables. Often in function optimization transformations are made to the variables such as random shifting. However, applying a random shift breaks the functional relationship between variables. In the context of a GP approach this makes no sense!

3. CARTESIAN GP AND EXPERIMENTAL PARAMETERS

Cartesian genetic programming (CGP) is a graph-based form of genetic programming [4]. The genotypes encode directed acyclic graphs and the genes are integers that represent where nodes gets their data, what operations nodes perform on the data, and where the output data required by the user is to be obtained. When the genotype is decoded, it can happen that some nodes are not referenced (i.e. they and their genes are 'non-coding'). We call the graph that results from this decoding, a phenotype. The genotype in CGP has a fixed length. However, the size of the phenotype (number of nodes) is variable. In this study, the inputs (or terminals) x_i, are generated randomly at the start of each evolutionary run. We generated ten real-valued constant in range [-1, 1]. The function set chosen for this study are defined over the real-valued interval [-1.0, 1.0]. The number of outputs is $n_o = d$, where d is the dimensionality of the optimization problem. Since the terminals and functions all return numbers in the interval [-1, 1] the program outputs, p_i also have values defined in this range. However the optimization functions are defined over a variety of intervals, which we denote by $[l_i, u_i]$. Thus the program outputs, p_i, need to be mapped to the intervals defined in the optimization problem, q_i. We do this using the set of linear mappings defined in equation Eqn. 6.

$$q_i = \frac{u_i - l_i}{2} p_i + \frac{u_i + l_i}{2}. \quad (6)$$

The primitive functions we used are the average of the two inputs, and the product of the inputs. In preliminary experiments we found these functions to work well. Unlike standard CGP, *three* mutation parameters were defined. A probability of mutating connections, μ_c, functions, μ_f and outputs, μ_o. As is standard in CGP a variant on a simple $1+\lambda$ evolutionary algorithm was used, where $\lambda = 4$ [4]. Thus in each population of five, one is the parent (promoted from the previous population) and the offspring are produced through mutation. In all experiments $\mu_c = 0.001$, $\mu_f = 0.002$, and $\mu_o = 0.03$. We set the genotype length to be 1000 nodes.

It is well known that CGP genotypes have many non-coding genes this leads to many offspring having the same phenotype as their parent. Thus, before we evaluate the fitness of the offspring we compare its output vector to the output vector of the parent, if it is identical we set its fitness to that of its parent. If it is not-identical we calculate its fitness and increment the variable that counts the number of fitness evaluations. In this way we can set a limit on the number of fitness evaluations.

4. RESULTS

Vesterstrom compared the performance of Differential evolution (DE), Particle Swarm Optimization (PSO) and an evolutionary algorithm (SEA) on a series of benchmark functions [6]. We compared the performance of CGP on these benchmarks under the same experimental conditions. We found that that for 15/20 benchmarks CGP attains the same of better (statistically significant) solutions than DE, while in 19/20 cases CGP attains the same or better solutions than PSO or SEA. Clearly it is a promising method.

5. CONCLUSIONS

We have described the application of CGP to function optimization. Such an approach, in addition to finding the optima of functions, allows functional relationships between variables to be found, thus potentially reducing the dimensionality of the problem and leading to a deeper understanding of the optimization problem. Clearly the work requires more experimental analysis and comparisons with other methods and also new kinds of benchmarks could be devised which show the advantages of the proposed method. We suggest that using GP to solve these kinds of problems opens up a new area of application for GP and also provides benchmarks whereby different GP techniques can be compared with one another.

6. REFERENCES

[1] Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A.: Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems. Applied Soft Computing 11, 5755–5769 (2011)

[2] Mallipeddi, R., Suganthan, P.N.: Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization. Tech. rep., Nanyang Technological University (2010)

[3] McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., Jong, K.A.D., O'Reilly, U.M.: Genetic programming needs better benchmarks. In: Proc. Genetic and Evolutionary Computation Conference (GECCO) 2012. pp. 791–798. ACM (2012)

[4] Miller, J.F. (ed.): Cartesian Genetic Programming. Springer (2011)

[5] Pujol, J.C.F., Poli, R.: Parameter Mapping: A genetic programming approach to function optimization. Int. J. of Knowledge-Based and Intelligent Engineering Syst. 12, 29–45 (2008)

[6] Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Evolutionary Computation, 2004. CEC2004. Congress on. vol. 2, pp. 1980 – 1987 (2004)

[7] Walker, J.A., Miller, J.F.: Changing the genospace: Solving GA problems with cartesian genetic programming. In: Proc. EuroGP. LNCS, vol. 4445, pp. 261–270. Springer (2007)

[8] Walker, J.A., Miller, J.F.: Solving real-valued optimisation problems using cartesian genetic programming. In: Proc. GECCO. pp. 1724–1730. ACM (2007)