Analytic Solutions to Differential Equations under Graph-Based Genetic Programming

Tom Seaton[1], Gavin Brown[2], and Julian F. Miller[1]

[1] Department of Electronics, University of York
[2] School of Computer Science, University of Manchester

Abstract. Cartesian Genetic Programming (CGP) is applied to solving differential equations (DE). We illustrate that repeated elements in analytic solutions to DE can be exploited under GP. An analysis is carried out of the search space in tree and CGP frameworks, examining the complexity of different DE problems. Experimental results are provided against benchmark ordinary and partial differential equations. A system of ordinary differential equations (SODE) is solved using multiple outputs from a genome. We discuss best heuristics when generating DE solutions through evolutionary search.

1 Introduction

Differential equations are ubiquitous throughout the natural sciences, modelling diverse systems from the harmonics of a violin to chemical concentrations in the blood stream. Widely applied finite difference and finite element routines can obtain accurate numerical approximations to DE solutions over prescribed domains [1]. However, the automatic derivation of analytic solutions to many high order or strongly non-linear DE problems remains challenging under standard deterministic approaches. Publicly available symbolic solvers apply analytic techniques such as separation of variables or symmetry reduction, but address specific classes of DE and require extensive supporting libraries for comprehensive support [2].

There has been some interest in the application of machine learning to DE since the advent of genetic algorithms. In this paradigm, the analytic solution can be considered analogous to a search objective, and the equation and any initial or boundary conditions to training data. In 1996 Diver evolved candidate solutions to simple ordinary differential equations (ODE), encoding solutions as strings [3]. Koza briefly addressed learning solutions to ODEs in his seminal work on problems for Tree Genetic Programming (GP) [4]. Cao later used an embedded genetic algorithm to tune parameters of a tree GP based solver [5]. More recently, Tsoulos and Lagaris set out fitness functions using a framework based on Grammar Guided GP [6] and applied them over a comprehensive collection of partial differential equations (PDE) and systems of ODE (SODE). We are also aware of a novel hybrid GP approach to solving DE implemented by Kirstukas et. al [7] for engineering applications.

Given the broad range of techniques historically applied, our goal is to identify general qualities which enable GP frameworks to efficiently explore DE search spaces. One defining characteristic of analytic solutions to differential equations is symbolic symmetry. Where solutions exist, application of the basic rules of differential calculus to trigonometric, polynomial or exponential functions naturally leads to repeated structure. In the present work, we postulate that DE have a higher density of analytic solutions under heuristic search in data-structures which reuse common functional elements. To explore this concept, we carry out an analysis using a variant on an established graph-based approach, Cartesian Genetic Programming (CGP) [8], across a set of benchmark DE. The technique is also examined under simple tree genetic programming, with a view to understanding whether representations reusing repeated structure have an advantage over these particular classes of problem.

Section 2 of this paper outlines some preliminaries of the CGP differential solver, including the specification of appropriate fitness functions and reproduction strategy. Section 3 analyses the corresponding search space composition and describes a method of enumerating DE solutions within CGP data structures. A set of experimental results are presented over benchmark DE in Section 4 and discussed in Section 5. The work concludes with general comments on using evolutionary frameworks for solving DE and a summary of outcomes.

2 CGP Implementation

2.1 Reproduction Strategy

The original implementation of Cartesian Genetic Programming can be classed as a strongly elitist method, since it uses a *1+λ* evolutionary strategy without crossover. The approach selected a single parent from each generation, promoting itself and λ mutated offspring. Our choice of this representation was motivated because the framework provides convenient reuse of previous elements through the evolution of more general directed acyclic graphs (DAG). For this initial analysis, we preclude techniques such as the use of modules [9] or automatically defined functions (see comment in Section 5). The CGP policy adopted in this paper employs a weak form of elitism shown in Figure 1.

```
1: $g \leftarrow 0$
2: Construct a random starting population of size $P$. Rank by fitness $F$.
3: **while** $g < g_{max}$ and $F > tol$ **do**
4:     Select a new parent with the best fitness.
5:     Promote the parent and $\lambda$ mutated offspring.
6:     Promote one offspring from each of the next $P-(\lambda+1)$ fittest individuals.
7:     Re-rank using the promoted population.
8:     $g \leftarrow g+1$
9: **end while**
```

Mutations are standard point operations on the CGP genotype and *tol* is some minimum error bound on individual fitness. For $\lambda = 1$ the method is random

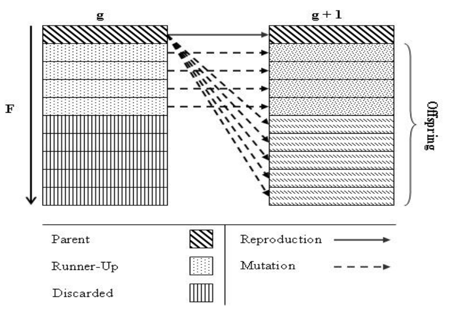


Fig. 1. Strategy for $\lambda = 5$, $P = 10$. The next generation contains the previous parent, five offspring of the parent and four offspring from the next best individuals.

search and $\lambda = P - 1$ is equivalent to conventional CGP. Heuristically, the trade off in selecting lower values of λ is greater diversity in the search, which reduces the risk of stagnation. More generally, the policy is related to a conventional $\mu+\lambda$ search but competes a larger proportion of offspring from the fittest parent. In preliminary experiments, $\lambda = P/2$ was found to be an adequate compromise for all the benchmark ODE and PDE addressed.

2.2 Fitness Functions for Differential Equations

Consider the problem of finding a particular closed analytic solution A to a bounded ODE or PDE, within a domain of interest spanned by the D-dimensional orthogonal basis set $x(x_1...x_D)$. A is an expression in x which

1. Satisfies the defining equality.
2. Meets all boundary or initial conditions.
3. Remains real and finite.

We adopt the approach of Tsoulos and Lagaris [6], evolving a model function $M(x)$, taking a weighted aggregate as the fitness $F(M)$. In general this has the form

$$F(M) = R(M) + \alpha\epsilon(M) \tag{1}$$

where R and ϵ are residual errors calculated for M across the equation and conditions respectively. α is an integer weighting parameter. Table 1 shows $F(M)$ for

Table 1. Fitness Functions for DE bounded on a line and a unit square

Differential Equation	$R(M)$	$\epsilon(M)$
$\frac{dy}{dx} = g(x,y), \quad y(x_0) = c$	$\sum_{i=1}^{N} \lvert M(x_i) - \frac{dM(x_i)}{dx}\rvert$	$\lvert M(x_i) - y(x_i)\rvert_{i=0}$
$\nabla^2\Psi = g(\Psi, x, y)$ $0 \leq x \leq 1$ $0 \leq y \leq 1$	$\sum_{i=1}^{N^2} \lvert M(x_i, y_i) - \nabla^2(M(x_i, y_i))\rvert$	$\sum_{i=1}^{N}(\lvert M(0, y_i) - \Psi(0, y_i)\rvert$ $+\lvert M(1, y_i) - \Psi(1, y_i)\rvert$ $+\lvert M(x_i, 0) - \Psi(x_i, 0)\rvert$ $+\lvert M(x_i, 1) - \Psi(x_i, 1)\rvert)$

an example 1st order ODE and 2nd order PDE. In the CGP solver, the differential terms are calculated by carrying out $O(N^D)$ centered difference approximations at uniformly sampled training points in the inner domain and $O(N(D-1)+1)$ along each bound. Centered difference approximations are evaluated using two sampling points separated by distance h, contributing error $O(h^2)$.

3 Problem Complexity and Search Space Analysis

3.1 Solving Differential Equations in Evolutionary Search

Abstractly, a DE solver under Graph-based GP searches through a finite, discrete set $\mathbf{\Omega}$ of representable expressions for members of a subset of solutions, $\omega \subset \mathbf{\Omega}$. The solution space ω contains all expressions from $\mathbf{\Omega}$ which are functionally equivalent to the analytic solution. The concept is illustrated in Figure 2 below. One method of defining the inherent difficulty of a DE problem is to consider the probability p_A of selecting an analytic solution, a member of ω, by blind random search. For convenience we work with $\frac{1}{p_A}$, terming this the 'unguided complexity' κ.

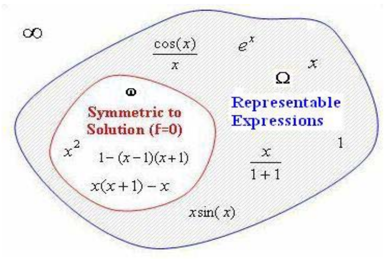


Fig. 2. Depiction of expressions in the search space of an ODE with solution x^2

Definition 1. *Unguided Complexity*

$$\kappa = \frac{1}{p_A} = \frac{|\mathbf{\Omega}|}{|\omega|}$$

such that κ is the number of candidates in the total search space per member of the solution set. A genotype representation of a problem where κ is large induces proportionally fewer optimal solutions and is combinatorially harder before the fitness landscape is considered.

3.2 Search Space Ω

Taking a standard case from Koza[4], consider a full bi-arity tree genotype of depth D. The tree is covered by $2^D - 1$ functions (interior vertices) and 2^D terminals (leaves). Hence for function and terminal sets of size f and T respectively, the number of syntactically distinct, labelled trees is given by

$$|\mathbf{\Omega_{TREE}}| = f^{2^D-1}\, T^{2^D} = T(fT)^{2^D-1} \tag{2}$$

For the CGP genotype, $\mathbf{\Omega_{CGP}}$ is instead defined by combinations of directed graphs. The closest comparison to the bi-arity tree in CGP is a feed-forward bi-arity genotype of the form shown in Figure 3. Here the bounding parameter on $\mathbf{\Omega_{CGP}}$ is the total genotype length, the number of genes C. When $C = 1$ the search space is $\mathbf{\Omega_{CGP}} = fT^2$. Generalising to permit connections to all previous nodes, the size of the total search space is

$$|\mathbf{\Omega_{CGP}}| = f^C \prod_{i=0}^{C-1} (T+i)^2 \tag{3}$$

or more transparently the factorial

$$|\mathbf{\Omega_{CGP}}| = f^C \prod_{i=0}^{C-1} (T+i)^2 = f^C \left(\frac{(T+C-1)!}{(T-1)!}\right)^2 \tag{4}$$

Comparing tree *depth* with CGP *length* is not straightforward, because they may imply a different number of nodes per individual. One method is to consider instead the maximum path length in each graph, such that $C = D$. From Equations 2 and 4, we then have

$$|\mathbf{\Omega_{CGP}}| < |\mathbf{\Omega_{TREE}}| \text{ given that } C, D > 4 \tag{5}$$

using any set of functions and terminals with $f > 1$ or $T > 1$. The feed-forward CGP applied in this experiment explores a smaller space of candidates than the full tree structure, as the maximum path length increases.

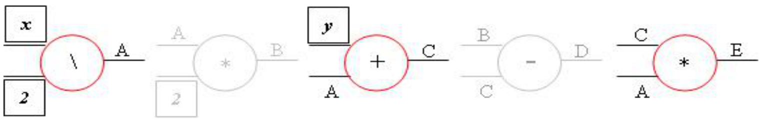


Fig. 3. Example 5 cell bi-arity genome from a single row CGP genotype. Cells 1,3 and 5 are connected to the output. Cells 2 and 4 are redundant.

3.3 Solution Space ω

With increasing genotype length, ω also expands to include increasingly complex symmetry. An alternative strategy to the challenge of enumerating ω directly for DE is to make an estimate by blind random sampling. Consider a number of independent GP runs R on a population size P. Classically the probability of success of an independent run under random search with no evolutionary mechanism is just the binomial product

$$1 - (1 - p_A)^{Pg} \tag{6}$$

$P \times g$ being the number of individuals created by generation g. Therefore a good estimate for p_A can be found by empirically fitting Equation 6 to a cumulative histogram of successes over all runs. Combined with Equations 2 and 4, the experiment gives us κ and ω for each DE problem. We apply this to a set of benchmarks in Section 4.

3.4 Parameter Space

To solve an ODE or PDE with genetic programming, the terminal and functions sets should be specified such that at least one group of elements can be drawn and ordered to give an expression equivalent to the desired analytic solution. Figure 4 illustrates how the unguided complexity can increase with additional functions and terminals. In this simple example, κ tends to grow roughly linearly with T and f. Interestingly, we note that the complexity when searching with both *log* and *exp* operators is lower than with a set precluding one or the other. In practice, the availability of the inverse operation introduces new equivalent solutions within ω and increases the probability of finding an analytic solution under random search. For simplicity, the following experiments use the *minimum* subset of functions under which the analytic forms of all benchmark problems addressed can be represented. Similarly, any constants required by the search are pre-seeded, rather than evolved dynamically. Division by zero is protected, returning one. The full function set included the operators ($+$ $-$ $*$ $/$ *sin cosine log exp*).

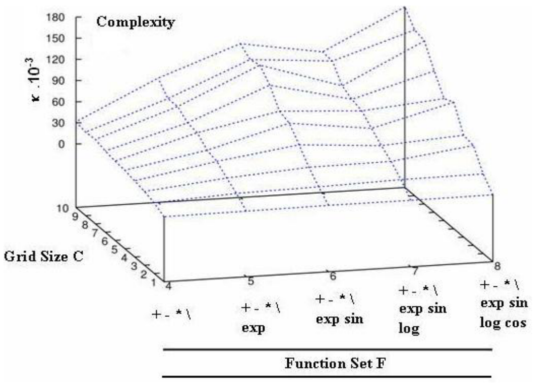


Fig. 4. The parameter space for a one dimensional ODE problem where $\frac{dy}{dx} = \frac{2y}{x}$, with initial condition $y(1) = 1$. $T = 2$ and $C = 4$).

4 Experiment

Exploring the effectiveness of different GP representations for solving DE requires a defined set of benchmark ODE and PDE. To allow ease of comparison, in the following sections we detail a selection of problems drawn from previous publications [3][6][7]. These problems cover a cross-section of different classes, order and domains. Table 2 below describes the parameters used throughout for each algorithm. Values for boundary weighting α and sampling rate N are optimised over the whole problem set. The maximum number of candidate trees or graphs was held constant between each set up ($P \times g_{max} = 10^7$). Candidates in tree GP were selected through a binary tournament strategy, with optimal population size $P = 1000$. Bloat control was introduced by constraining trees to a maximum of 150 nodes.

Table 2. Parameters for CGP and Tree Guided Search

Parameter	CGP	Tree
Population P	10	1000
Max Generations g_{max}	2000	20
Runs	500	500
Weighting α	100	100
Offspring λ	5	-
Sample Rate N	10	10
Mutation Rate	Point	2%
Crossover Rate	-	90%
Reproduction Rate	-	8%

4.1 Complexity and Performance against ODE

The ODE problem set chosen is summarised in Table 3. These consist of linear and non-linear problems for which closed form polynomial, trigonometric and exponential solutions exist. The above problems are chosen to test different aspects of the search algorithms. ODEs [3,9] have similar functional form in their solutions, but treat different equations. These problems should therefore have a comparable unguided complexity, but evaluate differently under the guided

Table 3. ODE Problem Set

No.	ODE	Domain	Conditions	Solution
1	$y' = \frac{2x-y}{x}$	$0.1 : 1.0$	$y(0.1) = 20.1$	$y = x + \frac{2}{x}$
2	$y' = \frac{1-ycos(x)}{sin(x)}$	$0.1 : 1.0$	$y(0.1) = \frac{2.1}{sin(0.1)}$	$y = \frac{x+2}{sin(x)}$
3	$y' = -\frac{1}{5}y + e^{-\frac{x}{5}}cos(x)$	$0.0 : 1.0$	$y(0.0) = 0$	$y = e^{-\frac{x}{5}}sin(x)$
4	$y' + ycos(x) = 0$	$0.0 : 1.0$	$y(0.0) = 1$	$y = e^{-sin(x)}$
5	$y' - \frac{2y}{x} = x$	$0.1 : 1.0$	$y(1) = 10$	$y = x^2ln(x) + 10x^2$
6	$y' + y^2 = 0$	$0.0 : 1.0$	$y(1.0) = 0.5$	$y = \frac{1}{1+x}$
7	$y'' = -100y$	$0.0 : 1.0$	$y(0) = 0, y'(0) = 10$	$y = sin(10x)$
8	$y'' = 6y' - 9y$	$0.0 : 1.0$	$y(0) = 0, y'(0) = 2$	$y = 2xe^{3x}$
9	$y'' = -\frac{1}{5}y' - y - \frac{1}{5}e^{-\frac{x}{5}}cos(x)$	$0.0 : 2.0$	$y(0) = 0, y'(0) = 1$	$y = e^{-\frac{x}{5}}sin(x)$
10	$y'' = 4y' - 4y + exp(x)$	$0.0 : 1.0$	$y(0) = 3, y'(0) = 6$	$y = e^x + 2e^{2x} + xe^{2x}$

search. ODEs [5,8,10] are examples of problems with solutions having more complex functional forms and dependency on several common sub-elements. Figure 5 contrasts the unguided complexity κ of the ODE problem set under tree GP and CGP. Over all known complexity estimates, blind random sampling using the CGP framework achieved a higher success rate than in the Tree framework, by factors ranging between 2 (ODE [4,6]) and 100 (ODE [8]). The solutions to the example ODE are more densely represented under a graph-based framework than in a tree-based representation.

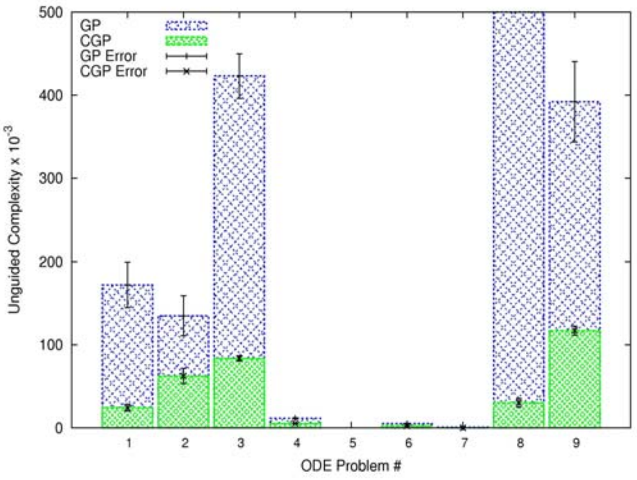
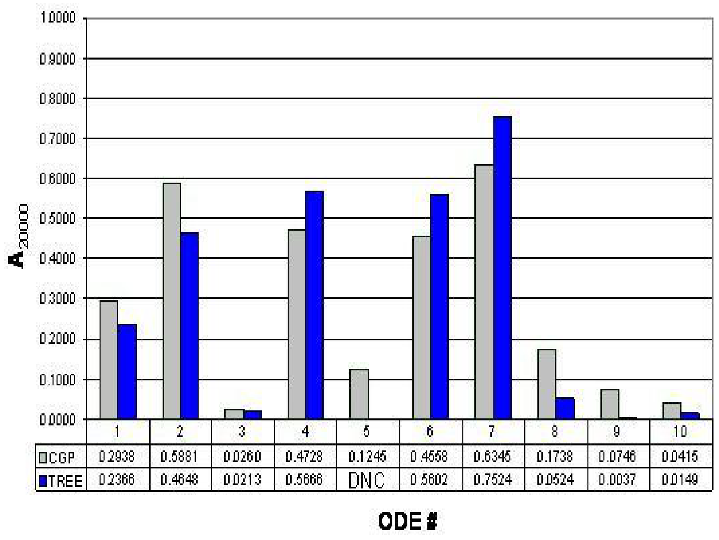


Fig. 5. Estimates of the unguided complexity, κ, on ODEs 1-9. ODES[5] and [10] did not converge under blind random search. All other complexity estimates are lower under CGP.

Figure 6 shows the performance under guided search. The results are presented using a convenient integral metric, such that

$$A_{I_{max}} = \frac{1}{10^7} \int_0^{I_{max}} h(I)dI \tag{7}$$

($I_{max} = 20000$) where A is the area underneath the probability of success curve $h(I)$, calculated as a discrete sum. $h(I)$ is the chance of having obtained an analytic solution after $I = P \times g$ candidate genomes, taken as an average from 100 runs of the solver. Overall performance for a given number of candidates is greater as $h \to 1$. The CGP solver converged on a representation of the analytic solution for all cases and performance showed good qualitative correlation with the complexity of each problem. Under guided search, the ODEs having a solution with a simple analytic form were solved more readily by the tree-based representation, but this did not converge for the high complexity case ODE 5. The best guided performance of CGP relative to the bi-arity tree GP occurred



	1	2	3	4	5	6	7	8	9	10
▨ CGP	0.2938	0.5881	0.0260	0.4728	0.1245	0.4558	0.6345	0.1738	0.0746	0.0415
■ TREE	0.2366	0.4648	0.0213	0.5666	DNC	0.5602	0.7524	0.0524	0.0037	0.0149

Fig. 6. Guided solver performance on ODE over 20000 candidate solutions, for CGP and Tree GP. Results are presented using an area metric (Equation 7).

Table 4. Elliptic PDE Problem Set

No.	PDE	Particular Solution
1	$\nabla^2\Psi(x,y) = 4$	$\Psi(x,y) = x^2 + y^2 + x + y + 1$
2	$\nabla^2\Psi(x,y) = -2\Psi(x,y)$	$\Psi(x,y) = sin(x)cos(y)$
3	$\nabla^2\Psi(x,y) = -(x^2 + y^2)\Psi(x,y)$	$\Psi(x,y) = sin(xy)$
4	$\nabla^2\Psi(x,y) + e^{\Psi} = 1 + x^2 + y^2 + \frac{4}{1+x^2+y^{2^2}}$	$\Psi(x,y) = log(1 + x^2 + y^2)$

for problems [1,2,5,8 and 10] which include repeat functional elements in their solutions.

4.2 Partial Differential Equations

An experiment was carried out to demonstrate proof of concept on partial differential equations. A collection of benchmark second order elliptic PDEs were solved using CGP across a Dirichlet bounded unit square, summarised in Table 4.[1] The full general function set was employed.

The full analytic solution was recovered in all cases. On average, the CGP algorithm took longest to recover the more complex functional forms of PDE[1] and PDE[4]. The convergence rate is thought to be slow for these problems because the fitness landscape is dominated by deep local minima near the true

[1] A complete specification of the PDE problems and their boundary conditions can be found in Tsoulos et. al. (2006) [6].

solution. Both solutions were often approximated by very fit combinations of nested sine functions, which pre-disposed the search to stagnate.

4.3 CGP and Systems of Differential Equations

An interesting quality of the CGP representation is the intended support for multiple outputs. Solutions are read from different starting nodes on the genotype and evolved simultaneously. We applied this aspect to a simple trigonometric SODE with repeated solutions of the form

$$\begin{aligned} y_1' &= cos(x),\ y_1(0) = 0 \\ y_2' &= -y_1,\ y_2(0) = 1 \\ y_3' &= y_2,\ y_3(0) = 0 \\ y_4' &= -y_3,\ y_4(0) = 1 \\ y_5' &= y_4,\ y_5(0) = 0 \end{aligned}$$

Outputs were obtained from the last n cells in the genome, expanding with one additional node for each extra ODE, repeating for $n > 5$. Figure 7 then shows the corresponding number of candidates I that must be evaluated for a 99% success probability from repeated runs. In this instance, the computational effort is approximately the same as solving a set of equivalent independent ODE. Under the graph-based representation, the new outputs are connected to existing partial solutions. This gives the appearance that solutions with a high degree of symmetry are grown easily, benefiting from 'cross-polination' along the genome.

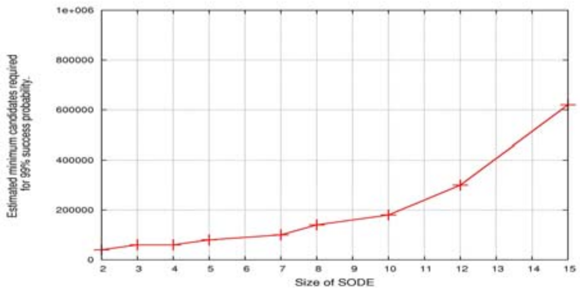


Fig. 7. Average number of individuals required to solve the test SODE, showing the increase with system size. Multiple outputs are read from the CGP genome.

5 Discussion

Section 4 demonstrated that GP search on standard DE problems can show faster convergence when working with representations which support repeated elements. In this instance, the mechanism we used to explore this was to apply a graph-based framework, but we would expect similar improvements when introducing techniques such as modules or ADFs. For the sample problems examined, more compact representations spanned a proportionally higher number of analytic solutions in the search space. It is worth noting that the difficulty of a DE under heuristic search is not strongly dependent on the form of the equation itself, but as in symbolic regression [11], on the length and diversity of the solution and also on boundary conditions.

Common to previous efforts, the main limitation of the approach is the requirement to define a sufficient function and terminal set. Applying prior knowledge of the boundary or initial conditions can provide useful indications of which subset to apply. Throughout all the problems, the boundary weighting condition α played a critical role in the search. It was found that defining a fitness function with strongly weighted boundaries ($\alpha = 100$) generally led to faster convergence on partial solutions, but naturally became dominated by these candidates as the weighting increased. A very low weighting ($\alpha < 1$) skews the population to include candidates which solve the general DE, but with a different functional form to the particular solution.

Differential equations are an interesting area for heuristic search techniques, both as an inductive tool and for engineering applications. In the latter, methods of seeding may be particularly useful, for example incorporating partial solutions based on empirical data. Another natural idea is to apply dimensional constraints, where the metric units of a problem are considered [10]. The approach could also be improved upon by including better techniques for evaluating candidate terms, such as automatic or symbolic differentiation [12].

6 Conclusions

The purpose of this initial work was to explore best heuristics for the evolution of solutions to differential equations under GP. We carried out an analysis of the search space and empirical performance of two GP solvers, conventional CGP and tree GP. It was illustrated that GP structures which automatically reuse common elements, such as graph-based representations, can show improved performance on target DE where solutions have repeated structure. Proof of concept was provided for a CGP solver on PDE and Systems of ODE. We further demonstrated that simple SODE were solvable using multiple outputs from a graph-based genotype and that this approach scaled efficiently with system size. A number of guidelines for solving DEs were inferred, including the selection of compact representations and strongly weighted boundary conditions.

References

1. Sobolob, S.L.: Partial Differential Equations of Mathematical Physics. Pergamen Press, Oxford (1964)
2. Abell, M.L., Braselton, J.P.: Differential Equations with Maple V. Academic Press, London (2000)
3. Diver, D.A.: Applications of genetic algorithms to the solution of ordinary differential equations. J. Phys. A: Math. Gen. 26, 3503–3513 (1993)

4. Koza, J.R.: Genetic Programming, on the Programming of Computers By Means of Natural Selection. MIT Press, Cambridge (1992)
5. Cao, H., Lishan, K., Chen, Y.: Evolutionary Modelling of Systems of Ordinary Differential Equations with Genetic Programming (2000)
6. Tsoulos, I., Lagaris, I.E.: Solving differential equations with genetic programming. Genet. Program. Evolvable Mach. 7, 33–54 (2006)
7. Kirstukas, S.J., Bryden, K.M., Ashlock, D.A.: A hybrid genetic programming approach for the analytical solution of differential equations. International Journal of General Systems 34, 279–299 (2005)
8. Miller, J.F., Thomson, P.: Cartesian Genetic Programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
9. Walker, J., Miller, J.F.: Investigating the performance of module acquisition in cartesian genetic programming. In: Genetic and Evolutionary Computation Conference, pp.1649–1655, 25-06 (2005)
10. Durrbaum, A., Klier, W., Hahn, H.: Comparison of Automatic and Symbolic Differentiation in Mathematical Modeling and Computer Simulation of Rigid-Body Systems. Multibody System Dynamics 7, 331–355 (2002)
11. Scmidt, M., Hod, L.: Comparison of Tree and Graph Encodings as Function of Problem Complexity. In: Genetic and Evolutionary Computation Conference, pp. 1674–1679 (2007)
12. Keijzer, M.: Scientific Discovery using Genetic Programming. PhD thesis, Department for Mathematical Modelling, Technical University of Denmark (2001)