Engineering Applications of Artificial Intelligence 55 (2016) 292–306

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Inference of compact nonlinear dynamic models by epigenetic local search

William La Cava [a,*], Kourosh Danai [a], Lee Spector [b]

[a] Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst, USA
[b] School of Cognitive Science, Hampshire College, Amherst, USA

ARTICLE INFO

Article history:
Received 3 December 2015
Received in revised form
25 June 2016
Accepted 19 July 2016
Available online 6 August 2016

Keywords:
System identification
Genetic programming
Dynamical systems
Differential equations
Symbolic regression

ABSTRACT

We introduce a method to enhance the inference of meaningful dynamic models from observational data by genetic programming (GP). This method incorporates an inheritable epigenetic layer that specifies active and inactive genes for a more effective local search of the model structure space. We define several GP implementations using different features of epigenetics, such as passive structure, phenotypic plasticity, and inheritable gene regulation. To test these implementations, we use hundreds of data sets generated from nonlinear ordinary differential equations (ODEs) in several fields of engineering and from randomly constructed nonlinear ODE models. The results indicate that epigenetic hill climbing consistently produces more compact dynamic equations with better fitness values, and that it identifies the exact solution of the system more often, validating the categorical improvement of GP by epigenetic local search. The results further indicate that when faced with complex dynamics, epigenetic hill climbing reduces the computational effort required to infer the correct underlying dynamics. We then apply the method to the identification of three real-world systems: a cascaded tanks system, a chemical distillation tower, and an industrial wind turbine. We analyze its solutions in comparison to theoretical and black-box approaches in terms of accuracy and intelligibility. Finally, we analyze population homology to evaluate the efficiency of the method. The results indicate that the epigenetic implementations provide protection from premature convergence by maintaining diversity in silenced portions of programs.

1. Introduction

A major goal of science is to characterize analytically the dynamic behavior of natural phenomena associated with biological, ecological, social, and economic systems, as well as the dynamics of artifacts such as wind turbines, robots, and aircraft. Dynamic behaviors are usually characterized by differential equations which in aggregate represent the dynamic model of the system. These dynamic models are the essence of the simulations that estimate/predict system behavior for policy decisions, design, optimization, control, and/or automation. This paper presents a method for construction of concise and mechanistically meaningful dynamic models from observations.

Dynamic models are preferably formulated according to first principles, to embody the knowledge of the process. However, first-principles models cannot often fully characterize the nonlinear dynamics of the process, as represented by process observations. In regress, first-principles models may be abandoned in favor of empirical models such as neural networks (Narendra and Parthasarathy, 1990; Gregorčič and Lightbody, 2008), linear or nonlinear autoregressive moving average (ARMAX) models (Ljung, 1999; Billings, 2013), or others (Ni et al., 1996; Sadollah et al., 2015), that have the structural flexibility to accommodate the measured process observations. Although these empirical models provide an effective basis for estimation/prediction, they have two major drawbacks. One is their 'black-box' format which obscures the knowledge of the process acquired through adaptation. The second is their case-specificity which makes them potentially deficient in representing the process under conditions (inputs) not encompassed by the measured observations. To remedy the black-box nature of these empirical models, dynamic models consisting of differential equations can be defined in algebraic form by symbolic regression (Gray et al., 1998; Cao et al., 2000; Bongard and Lipson, 2007), wherein both the structure (topology) and parameters (constants) are inferred from measured observations. Since these symbolic models are intelligible, they have the capacity to elucidate the process physics. Symbolic regression is typically conducted using genetic programming (GP) (Koza, 1992), which is a bio-inspired machine learning technique that

* Corresponding author.
E-mail address: wlacava@umass.edu (W. La Cava).

http://dx.doi.org/10.1016/j.engappai.2016.07.004
0952-1976/

constructs candidate models from mathematical building blocks and proceeds with selection, recombination and mutation over several generations before converging on a model that best fits the process observations.

In comparison to system identification methods that presume fixed model structures, symbolic regression can be computationally expensive because of its expanded search space. Furthermore, when guided solely by an error metric, it can yield unwieldy equations that are elusive to physical interpretation. To remedy these shortcomings, this paper introduces a new method of symbolic regression that fine-tunes candidate model structures by local search (La Cava et al., 2015). This fine tuning is enabled by the addition of an epigenetic layer for selection of program components (consisting of variables and instructions) to be included in the model. The incorporation of this epigenetic layer is motivated by two hypotheses: first, that the benefits of epigenetic regulation observed in biology may confer analogous improvements on GP systems; and second, that generalized local search methods enabled by epigenetics may improve the ability of GP to find correct model structures.

As to the first hypothesis, despite the highly regulated nature of biological genes, the role of epigenetics in regulating gene expressions is traditionally ignored in GP (with some exceptions, e.g. (Ferreira, 2001)). However, epigenetic processes may provide several evolutionary benefits. For example, because epigenetic processes allow the underlying genotype to encode various expressions and lead to neutral variation through crossover and mutation of non-coding segments, they may allow populations to avoid evolutionary bottlenecks or let them respond to changing evolutionary pressures (Jablonka and Lamb, 2002). Also, because they provide for phenotypic plasticity that enables gene expression to change in response to environmental pressure (Dias and Ressler, 2013), they may allow gene expression adaptations to be inherited in offspring without explicit changes to the genotype. This property legitimizes, via epigenetic processes, once discredited ideas of Lamarck pertaining to the inheritability of lifetime adaptations (Jablonka and Lamb, 2002; Holliday, 2006).

Regarding the second hypothesis, although local search methods have been developed and integrated into evolutionary algorithms (Gruau and Whitley, 1993; Whitley et al., 1994; Jeong and Lee, 1996; Ross, 1999; Giraud-Carrier, 2002), especially in genetic algorithms (GAs) through prescribed changes to the genotype, the role of structure optimization in symbolic regression is typically left to the GP process. Aside from some recent developments (Arnaldo et al., 2014), local search is traditionally conducted at the genome level. More generic local search methods, like tree snipping (Bongard and Lipson, 2007), focus on improving secondary metrics like size or legibility, whereas the traditional search methods, like stochastic hill-climbing (Bongard and Lipson, 2007), linear (Iba and Sato, 1994) or non-linear regression (Topchy and Punch, 2001) are confined to constant optimization. Although these local search methods improve symbolic regression performance, they cannot aid the search for program topology.

Epigenetics, on the other hand, provide a natural basis for performing local search at the structural (i.e., program topology) level. Motivated by this benefit of epigenetics, we introduce in this paper an epigenetics-enabled GP system to conduct topological optimization of programs at the level of gene expression. The contributions of this method are twofold: first, it introduces a generic method of topological search of the space of individual genotypes via modifications to gene expression. Second, it improves programs without affecting the genotype and without discarding the acquired knowledge gained through evolution, thereby lowering the risk of premature convergence observed in previous studies (Whitley et al., 1994). These contributions are achieved by conducting local search on the epigenome rather than the genome and making these adaptations inheritable via evolutionary processes.

The proposed Epigenetic Linear Genetic Programming (ELGP) method is tested on a large array of data generated from nonlinear ordinary differential equations (ODEs), as well as from three real-world processes, to evaluate the quality of its solutions. The paper is organized as follows. We formulate in Section 2 the identification problem and describe in Section 3 the ELGP method and its application to inference of dynamic models. We also review the relevant work in the context of GP and nonlinear dynamics modeling in Section 4. We then present the experimental analysis of different epigenetic implementations on a series of increasingly complex problems in Section 5. We begin by testing the method on a large set of data obtained from simulated nonlinear ODEs in different engineering fields, in order to illustrate its breadth of application. We then perform identification on hundreds of randomly constructed nonlinear systems, varying in complexity and dimensionality, to evaluate the scalability of the method in comparison to traditional GP approaches. Finally, we apply the ELGP method to three real-world problems, including the identification of (1) a benchmark cascaded tanks system (Wigren and Schoukens, 2013), (2) a chemical distillation tower, and (3) an industrial wind turbine. The results are presented in Section 6 and include comparisons of ELGP's performance in relation to other linear and nonlinear identification methods. We finish this discussion with an analysis of population diversity to study how gene expression evolves for each ELGP implementation.

2. Problem statement

The underlying assumption of symbolic regression is that there exists an analytical model of the system that would generate the measured observations $y(t_k)$ at the sample times $t_k = t_1, \ldots, t_N$ under the input, $\mathbf{u}(t)$, as

$$y(t_k, \mathbf{u}) = \hat{y}(t_k, M^*(\mathbf{x}, \mathbf{u}, \boldsymbol{\Theta}^*)) + \nu; \quad k = 1, \ldots, N \tag{1}$$

where $\hat{y}$ is the model output, ν represents measurement noise in y, $\mathbf{x} = [x_1, \ldots, x_n]^T$ is the vector of state variables, and $M^*(\mathbf{x}, \mathbf{u}, \boldsymbol{\Theta}^*)$ is the correct model form embodied by the correct parameter values $\boldsymbol{\Theta}^*$, written M^* hereafter for brevity. In the search for the correct model form M^*, GP typically attempts to solve the problem

$$\text{minimize } f(M) \text{ subject to } M \in \mathfrak{S} \tag{2}$$

where $\mathfrak{S}$ is the space of possible models M, and f denotes a minimized fitness function. Given that it is impractical to exhaustively search $\mathfrak{S}$, the model found to minimize $f(M)$ may only be locally optimal. For practical purposes it is assumed that a suboptimal model can nevertheless fulfill the purpose of adequately representing the process, as depicted by the measured observations.

A common choice for estimating a candidate model output $\hat{y}(\widehat{M})$ is numerical integration or simulation of the state variables, i.e. the "output error" method (Ljung, 1999). However, given the sensitivity of simulation to different model structures (La Cava and Danai, 2015) and the computational cost of numerical integration, the alternative approach of algebraically estimating candidate model outputs is preferred for symbolic regression (Bongard and Lipson, 2007; Schmidt and Lipson, 2009). In the algebraic approach, un-measured states, denoted $\tilde{\mathbf{x}}$, are estimated from measurements via numerical differentiation together with smoothing functions. In the case of first-order differential equations with unmeasured state derivatives, the target is estimated numerically as $y(t_k, \mathbf{u}) = \tilde{x}$, such that the prediction error of a candidate model has the form

$$\epsilon(t_k) = y(t_k) - \hat{y}(t_k, \widehat{M}(\mathbf{x}, \mathbf{u}, \widehat{\boldsymbol{\Theta}})) = \tilde{x}(t_k) - \hat{\tilde{x}}(t_k) \tag{3}$$

The fitness metric f in Eq. (2) for individuals is often defined using the mean absolute error (MAE) or mean squared error (MSE), although some prefer using the correlation coefficient due to its insensitivity to linear scaling (Keijzer, 2003; Kommenda et al., 2013). We use a fitness metric (La Cava and Danai, 2015) designed to minimize error and maximize correlation so that both the prediction error and the closeness of shapes defined by the coefficient of determination (R^2), are accounted for in the results. This fitness metric takes advantage of the covariance comparison afforded by R^2 and avoids the need for post-hoc linear scaling of the solutions which could lead to increased model complexity. For target y and output $\hat{y}$, f is defined as:

$$f = \frac{1}{N}\sum_{k=1}^{N} |\epsilon(t_k)|/R^2(y, \hat{y}) \tag{4}$$

$$R^2 = \frac{\left(\mathrm{cov}(y, \hat{y})\right)^2}{\mathrm{var}(y)\mathrm{var}(\hat{y})} \tag{5}$$

3. Epigenetic Linear Genetic Programming (ELGP)

In symbolic regression, the search for candidate models is conducted by GP, wherein a population of computer programs, consisting of variables and instructions that produce models of the process, are evolved. Mathematical building blocks compose the genotype of each program that is optimized by an evolutionary algorithm. The operation steps of ELGP,[1] outlined in Fig. 1, start with randomly constructed programs that comprise an initial population. The model outputs generated from these programs are evaluated with respect to the training data. Depending on the variant of ELGP as defined in Section 3, the population then undergoes some form of epigenetic adaptation. Afterwards, the population undergoes selection, recombination and mutation, as in standard GP, to produce an updated population. The process repeats until an adequate solution is produced.

The ELGP method has two salient features that improve its performance: (i) its use of linear, stack-based programs to represent equations, and (ii) its incorporation of local search of the model structures space to improve the candidate model's fitness and to reduce its complexity. These features have been shown to bolster traditional GP's performance on several benchmark regression problems in terms of the conciseness of the developed models, their fitness, and efficiency of the search (La Cava et al., 2014, 2015). The effectiveness of these features is evaluated here in construction of nonlinear dynamic models.

3.1. GP representation

An innovation of the proposed ELGP method is its utilization of stack-based representation (Perkis, 1994; Spector and Robinson, 2002) to accommodate the introduction of epigenetics. In this representation, programs are encoded as post-fix notation, linear genotypes. This stack-based GP system is advantageous because it guarantees syntactic validity for arbitrary sequences of instructions. This property allows instructions to be silenced or activated in a genotype without invalidating the program's ability to execute, in contrast to tree-based representations that can become syntactically invalid due to changes to instructions and literals.

[1] source code available from http://www.github.com/lacava/ellen.

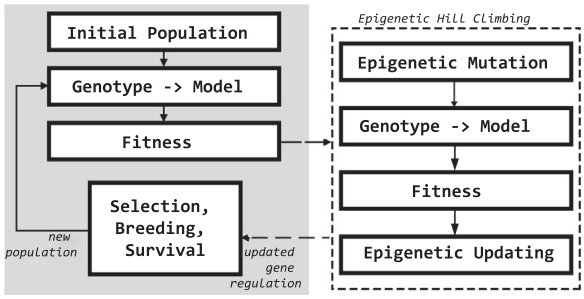

Fig. 1. Block diagram of ELGP. The typical GP steps are shown on the left. After fitness evaluation and before selection, the population undergoes an iteration of epigenetic hill climbing, represented by the block on the right.

The syntactic robustness of the stack-based approach is achieved mainly by ignoring the execution of instructions that have an arity larger than the current size of the stack. For example, if a $+$ operator attempts to execute and there is only one element on the stack, it does nothing. Furthermore, we base a program's behavior only on the top element of the stack after execution which allows programs to contain unused arguments. These two rules are the key to accommodating diverse program syntax. According to this flexibility, for instance, the genotypes of the following three programs $\mathbf{i_1}$, $\mathbf{i_2}$ and $\mathbf{i_3}$ will produce the identical model $(x_1 + x_2)$:

$$\begin{aligned}
\mathbf{i_1} &= [\,x_1\ \ x_2\ \ +\,] \Rightarrow M_1\text{: } (x_1 + x_2)\\
\mathbf{i_2} &= [\,x_1\ \ x_2\ \ +\ \ -\ \ *\ \ /\,] \Rightarrow M_2\text{: } (x_1 + x_2)\\
\mathbf{i_3} &= [\,u\ \ +\ \ x_1\ \ /\ \ x_1\ \ x_2\ \ +\,] \Rightarrow M_3\text{: } (x_1 + x_2)
\end{aligned} \tag{6}$$

The executions of $-$, * and $/$ in $\mathbf{i_2}$ are ignored due to insufficient stack size, and in $\mathbf{i_3}$, the last element of the executed stack, $(x_1 + x_2)$, is taken as the model. A step-by-step execution of program $\mathbf{i_3}$ is given in Fig. 2 to illustrate the procedure.

3.2. Epigenetic learning and evolution

We introduce epigenetic information into the GP representation by including an on/off marker on each element in an individual's genotype. This corresponding sequence of on/off markers is referred to as an epigenome. When evaluated together, the expressed program, i.e. model, is produced by executing instructions that are **on** (active) and ignoring the instructions that are **off** (inactive). In this light, one can see that the non-coding genes (known as introns (Wu and Lindsay, 1995) ignored in programs $\mathbf{i_2}$ and $\mathbf{i_3}$ in Eq. (6) provide local solutions to explore in the search space, making it possible to alter the topology and values of the resultant model. For example, program $\mathbf{i_3}$ can admit several

$\mathbf{i_3} = [\ u\ \ +\ \ x_1\ \ /\ \ x_1\ \ x_2\ \ +\ ]$

program execution	stack
1. (u) : push u	$[\,u\,]$
2. $(+)$: ignore (insufficient arguments)	$[\,u\,]$
3. (x_1) : push (x_1)	$[\,u\ \ x_1\,]$
4. $(/)$: pull $(x_1), (u)$; push (u/x_1)	$[\,(u/x_1)\,]$
5. (x_1) : push (x_1)	$[\,(u/x_1)\ \ x_1\,]$
6. (x_2) : push (x_2)	$[\,(u/x_1)\ \ x_1\ \ x_2\,]$
7. $(+)$: pull $(x_2), (x_1)$; push $(x_1 + x_2)$	$[\,(u/x_1)\ \ (x_1 + x_2)\,]$
$\rightarrow$ return last element on stack	$M_3 : (x_1 + x_2)$

Fig. 2. Stack-based execution of GP program $\mathbf{i_3}$ from Eq. (6). Arguments are pushed to the stack and operands (*, $+$, etc.) pull arguments from the stack, perform an operation, and push the result. Operands without sufficient arguments are ignored, and the final element on the stack at the end of execution is returned as the model.

models via epigenetic transformations, including

$$\mathbf{i_3} \rightarrow \mathbf{i_3}' = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ u & + & x_1 & / & x_1 & x_2 & + \end{bmatrix} \Rightarrow M_3' \colon (u + x_2)$$

$$\mathbf{i_3} \rightarrow \mathbf{i_3}'' = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ u & + & x_1 & / & x_1 & x_2 & + \end{bmatrix} \Rightarrow M_3'' \colon (u/x_1)$$

$$\mathbf{i_3} \rightarrow \mathbf{i_3}''' = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ u & + & x_1 & / & x_1 & x_2 & + \end{bmatrix} \Rightarrow M_3''' \colon (u/x_1 + x_2) \quad (7)$$

Similarly, program $\mathbf{i_2}$ in Eq. (6) admits the models $(x_1 + x_2)$, $(x_1 - x_2)$, $(x_1 * x_2)$, and (x_1/x_2) via epigenetic transformations.

During the ELGP process depicted in Fig. 1, the epigenetic markers are initialized randomly (in the initial population) with a probability of being active. We use 50% as the initial probability for the experimental studies in Section 5, chosen according to a previously conducted parametric study (La Cava et al., 2014). The extent to which epigenetic information is learned and inherited is a research question that we address by exploring different implementations. The simplest topological search method within ELGP is Ep1M, which mutates the epigenetic layer of each individual each generation; the hill climber in Fig. 1 is skipped. Thus for Ep1M, epigenetic mutations face only evolutionary pressures. In contrast, the epigenetic hill climbing (EHC) cases EHC1, EHC5 and EHC10 use the epigenetic information explicitly to improve individuals each generation (the EHC is described in Section 3.2.2). The three methods execute one, five and ten iterations of EHC each generation, respectively. Two control methods, Base and Ep0, are used for comparison. In the Base case, individuals are represented as basic genotypes as in Eq. (6). The Ep0 case acts like Base but with half of the genes in the initial code permanently silenced. As such Ep0 accounts for the effect that smaller active programs and explicit introns might have. Neither Base nor Ep0 use the right half of the system in Fig. 1 (i.e. the program never enters epigenetic mutation).

3.2.1. Epigenetic mutation

Whitley et al. (1994) introduced Lamarckian updating to GAs by conducting local search of the bit strings within 1 Hamming distance of the current bit string. In theory it would be possible to treat the epigenome as a bit string and proceed similarly. However, the cost of GP fitness evaluations may render this approach intractable. Instead, each generation, the epigenome is uniformly mutated with a probability of 10% at each gene. The mutation flips the binary value of the epigenome at the gene, thus activating or silencing that gene. The operation is uniform with respect to the number of instructions. Epigenetic mutation is illustrated in Fig. 3 to show how these epigenetic changes can result in significant topological changes to the resultant models. For the EHC1, EHC5 and EHC10 implementations, the epigenetic mutation is followed by hill climbing, described next.

3.2.2. Epigenetic hill climbing

In order to mimic the acquisition of lifetime learning by epigenetic adaptation, the EHC implementations evaluate epigenetic changes $\mathbf{i} \rightarrow \mathbf{i}'$ to determine whether individuals should be updated. At each iteration of epigenetic mutation, EHC1, EHC5 and EHC10 test the changes to the model for acceptance. Epigenetic changes to an individual are kept only if the fitness is improved or not changed, i.e. $f_{\mathbf{i}'} \leq f_{\mathbf{i}}$ (fitness f is being minimized).

In addition, we break fitness ties by preferring less complex equations. Model complexity can be represented by several approaches. For example, one can count the number of nodes in the parse tree, calculate the order of a Chebyshev polynomial fit to the model's output (Vladislavleva et al., 2009), or recursively aggregate the complexity of sub-expressions (Kommenda et al., 2015). Here, we account for model complexity by assigning component function nonlinearities to genotype components (Smits and Kotanchek, 2005). The complexity $C_{\mathbf{i}}$ of program $\mathbf{i}$ with active genotype $\mathbf{g_a} = \left[g_{a_1} \ldots g_{a_\ell} \right]$ is defined as $C_{\mathbf{i}} = \sum_{q=1}^{\ell} c(g_{a_q})$, where component function nonlinearities (Smits and Kotanchek, 2005) are defined as

$$c(g_a) = \begin{cases} 4\text{: } (g_a = \log) \vee (g_a = \exp) \\ 3\text{: } (g_a = \sin) \vee (g_a = \cos) \\ 2\text{: } (g_a = /) \vee (g_a = \sqrt{()}) \\ 1\text{: otherwise} \end{cases} \quad (8)$$

Lower-complexity programs with equivalent fitness are accepted, giving the condition

$$pass = (f_{\mathbf{i}'} < f_{\mathbf{i}}) \vee \left((f_{\mathbf{i}'} = f_{\mathbf{i}}) \wedge (C_{\mathbf{i}'} < C_{\mathbf{i}}) \right) \quad (9)$$

If the epigenetically mutated individual $\mathbf{i}'$ does not pass Eq. (9), the changes are discarded and $\mathbf{i}$ is kept in the population. Otherwise $\mathbf{i}$ is replaced with $\mathbf{i}'$.

3.2.3. Epigenetic inheritance

A key feature of ELGP is the inheritance of epigenetic values throughout the evolutionary process. During crossover, the epigenetic values of the parent genes are kept intact such that the child receives the epigenetic states of the genes it has inherited. If a new gene is introduced via genetic mutation, that gene has the same probability of being active as the initial genes of the population (50%, in the current study).

4. Related work

There has been some work to incorporate epigenetic learning into GP, notably by Tanev and Yuta (2008). In that case the focus was to model histone modification through a double cell representation as demonstrated in a predator-prey problem. Unlike our approach, Tanev did not treat lifetime epigenetic modifications as inheritable, as is supported by recent studies in biology (Turner,

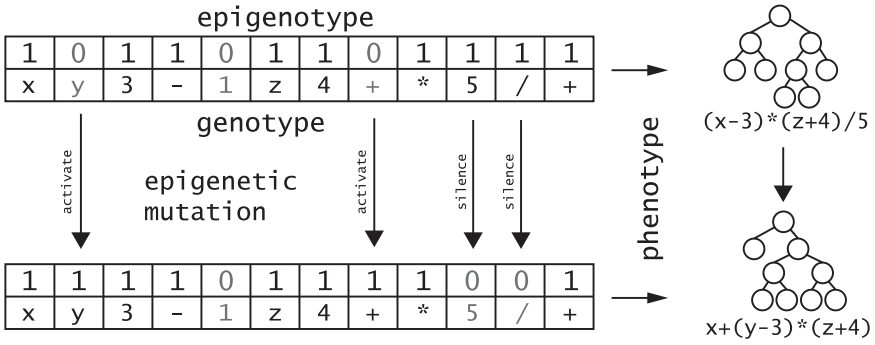

Fig. 3. Illustration of an epigenetic mutation applied to a GP program. The mutations result in topological changes to the model (phenotype), shown on the right.

2000; Kaati et al., 2002; Dias and Ressler, 2014). There have also been a number of studies on the effects of non-coding segments in GP, some of which have found that the structural presence of introns protect genotypes from destructive crossover operations (i.e., operations that produce children less fit than their parents) (Nordin et al., 1995; Wu and Lindsay, 1995; Brameier and Banzhaf, 2007). Non-coding segments were found to be useful in evo-devo for evolution of arbitrary shapes as well (Fontana, 2011). In each of these studies, introns were declared explicitly or measured during evolution, rather than being actively manipulated by the system itself as in ELGP. Our preliminary study of epigenetic initialization finds rates of beneficial crossover to be the highest with the probability set to 50% (La Cava et al., 2014).

In addition, several GP systems use similar stack-based or linear genome representations, such as PushGP (Spector and Robinson, 2002), Push-forth (Keijzer, 2013) and Gene Expression Programming (Ferreira, 2001), that could trivially implement the epigenetic layer incorporated in the ELGP method. Similarly, there are methods that leverage neutrality (i.e., different genotypes with the same fitness) by creating a genotype - phenotype mapping; e.g., Cartesian GP (Miller and Thomson, 2000) and Binary GP (Banzhaf, 1994). Developmental approaches to GP (Koza et al., 1997) also make use of genotype – phenotype mappings. Our goal with ELGP is to incorporate local search of gene expression as a viable, generic GP extension that does not require large changes to implement. As mentioned earlier, there are a plethora of studies on local search methods for improving GP by Lamarckian or Baldwinian means, yet very few have considered these changes to occur at the epigenetic level instead of the genotype level. A notable exception is Multiple Regression GP (Arnaldo et al., 2014), in which parameter values are implied at each node location and updated by linear regression. Still, the tangible improvements brought about by this and most other local search methods for symbolic regression are achieved by parametric, rather than topological, search.

Several methods based on GP have been proposed for modeling nonlinear dynamic systems, including ODE model structures (Gray et al., 1998; Cao et al., 2000; Bongard and Lipson, 2007; Schmidt and Lipson, 2009) and GP-NARMAX models (Rodríguez-Vázquez and Fleming, 2004; Rodriguez-Vazquez et al., 2004). GP populations are often evolved with multi-objective methods like SPEA2 (Zitzler et al., 2001) and NSGA-II (Deb et al., 2000) to pressure model size. Unlike ELGP, these techniques for compact modeling focus on changes to the core GP algorithm (selection and fitness evaluation). In this regard, ELGP could be readily applied in those proposed frameworks. The multiobjective framework we use in our experiments is that proposed in Schmidt and Lipson (2011), as discussed in Section 5.1.

More broadly, GP is one approach to nonlinear system identification, among many others. A common approach embodied by Hammerstein-Weiner and nonlinear auto-regressive modeling with exogenous inputs (NARX) is to combine a chosen nonlinear transformation (or transformations) with a linear model. Although the use of a nonlinear estimator can increase the capacity compared to ARX modeling, per say, in both cases the structure of the nonlinearity must be specified beforehand, contrary to symbolic regression. These approaches also produce complex models that can obfuscate intuitive explanations of their predictive power. To remedy this, greedy structure selection methods have been proposed for nonlinear polynomial models (Haber and Unbehauen, 1990), notably for the NARMAX model using the orthogonal least squares (OLS) (Chen et al., 1989; Billings, 2013). GP methods have also been proposed to optimize the structural identification of OLS models (Madar et al., 2004). The goal of ELGP is to improve the ability of GP representations to produce intelligible model structures, which in turn can be applied to auto-regressive representations (La Cava et al., 2015) and/or coupled with a desired parameter estimation strategy (Iba and Sato, 1994; Topchy and Punch, 2001; Kommenda et al., 2013).

5. Experimental methods

We describe in this section the evolutionary framework to which ELGP is applied and the settings that are used to conduct the experiments, followed by a description of the set of problems that are used to compare the performance of each GP treatment. Section 5.1 describes the algorithms used to perform selection and search operations within GP, which build upon previous symbolic regression research. In Section 5.2, we describe implementation optimizations related to efficiently performing hill climbing on epigenetically mutated programs. Finally in Section 5.3 we present the set of problems on which ELGP is evaluated, which include simulated ODEs from various fields, randomly constructed ODEs that vary in complexity, and three real-world nonlinear dynamics modeling applications.

5.1. Evolutionary algorithm

Several state-of-the-art symbolic regression tools leverage Pareto optimization for selection and survival (Smits and Kotanchek, 2005; Schmidt and Lipson, 2009), and our preliminary tests (not reported here) confirm that the age-fitness Pareto survival method (Schmidt and Lipson, 2011) outperforms traditional GP (Koza, 1992) on several problems. In an effort to demonstrate ELGP on a high-performance configuration, we use age-fitness Pareto survival as the evolutionary algorithm in our experiments. In this scheme, each individual is assigned an age equal to the number of generations since its oldest ancestor was created. Each generation, a new individual is introduced to the population as a means of random restart. Selection for breeding is random, and during breeding a number of children equal to the overall population size is created. At the end of each generation, environmental selection is conducted according to the Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler et al., 2001) to reduce the size of the set P comprising the current population plus the newly created individuals down to the original population size N. Note that the hypervolume method from NSGA-II (Deb et al., 2000) could also be used for this task, although previous work suggests that SPEA2 performs better in low dimensions (Zitzler et al., 2001).

SPEA2 uses two measures to perform this reduction of P: (1) Pareto strength of an individual, $S(\mathbf{i})$, which is the number of individuals equal to or dominated by $\mathbf{i}$, divided by $|P|+1$, and (2) a density estimate $D(\mathbf{i}) < 1$, based on the inverse of the distance to the k-th nearest neighbor (Silverman, 1986) of $\mathbf{i}$ in objective space (in this case the objectives are normalized between zero and one). These metrics are used to define a fitness value $F(\mathbf{i})$ that combines the total strength of the individuals $\mathbf{j} \in P$ that dominate $\mathbf{i}$, denoted as $\mathbf{j}<\mathbf{i}$, with density estimate $D(\mathbf{i})$, as

$$F(\mathbf{i}) = \sum_{j \in P, j<i} S(\mathbf{j}) + D(\mathbf{i}) \tag{10}$$

Every nondominated solution is first copied to the new population. If the new population size is smaller than N, individuals are added in order of lowest $F(\mathbf{i})$. If the population is larger than N, signifying that there are more than N nondominated solutions, individuals are removed iteratively based on $D(\mathbf{i})$. In the latter scenario, the use of $D(\mathbf{i})$ for selection helps preserve spread of solutions along the Pareto front.

A uniform alternation crossover operator is used to produce two children from two parents, as in Spector and Helmuth (2013). The mutation operator is applied uniformly to the chosen parent with a probability of 2.5% at each gene. If a constant gene is picked

Table 1
ELGP system settings as applied to the Textbook ODE problems.

General settings	Value
Population size	1000
Crossover/mutation	80/20%
Program length limits	[3, 50]
ERC range	[−10,10]
Termination criterion	2.5E10 point evals or $f < 1.0E-6$
Trials	100
Function set	{+, −, *, /, sin, cos}

for mutation and ephemeral random constants (ERCs) are being used, the constant is perturbed by Gaussian noise with standard deviation equal to half the magnitude of the constant. Otherwise the instruction is mutated to a randomly chosen gene.

In order to optimize the parameters (constants) of the models, one iteration of stochastic hill climbing is conducted on model parameters each generation. The hill climber perturbs all constant values in the active genotype by Gaussian noise with a standard deviation equal to 10% of the value of the constant. These changes are kept if they result in a better fitness for the individual. This method of constant optimization is chosen due to its lightweight nature compared to least-squares approaches.

Each trial was allocated a maximum number of point evaluations, i.e., gene executions, to normalize for the different program sizes among methods. A GP run will exit early if the fitness condition $f < 10^{-6}$ is achieved before the designated number of point evaluations has been reached. We observed this fitness termination condition to be sufficient for reaching exact solutions for the problems studied here.

5.2. Optimizations

The following optimization provisions are applied to ELGP in order to reduce the number of point evaluations required to evaluate the fitness of an individual that has undergone epigenetic mutation. The majority of run-time in most GP systems (including ours) is devoted to fitness evaluation. This motivates reduction of the number of point evaluations required.

Fitness escape. EHC requires additional fitness evaluations in order to determine whether the prescribed epigenetic changes will be kept. Given that the fitness $f_{\mathbf{i}}$ of program $\mathbf{i}$ cannot decrease with the evaluation of more fitness cases $k_{1\ldots n}$, evaluation of the epigenetically mutated individual $\mathbf{i}'$ can be halted if at any point $f_{\mathbf{i}'}(1\ldots k_j) > f_{\mathbf{i}}(1\ldots k_n)$ for $1 \le j < n$. This allows $\mathbf{i}'$ to be discarded before its fitness is fully evaluated because it is guaranteed to be worse than $\mathbf{i}$. Since fitness is always equal to or larger than MAE (see Eq. (4)), the halt condition can be defined conservatively using the mean absolute error (MAE) of $\mathbf{i}'$ and the fitness of $\mathbf{i}$ as

$$halt = \frac{1}{N}\sum_{k=1}^{j} |y(t_k) - y(t_k, M_{\mathbf{i}'})| > f_{\mathbf{i}} \tag{11}$$

Stack tracing. In GP tree representations, the output of a node in the program typically depends only on the outputs of its child nodes (and those children's children and so forth). We can say conservatively with ELGP representations that no instruction in the stack is dependent on an instruction to its right. Therefore, when a gene is silenced or activated, only the outputs of the genes to its right in the genotype are affected, hence only part of the program needs to be reevaluated. To avoid repeated instruction evaluations during epigenetic hill climbing, we save the intermediate program outputs of each gene, and after epigenetic mutation reevaluate only those genes to the right of the left-most location of mutation.

Saving the stack outputs is a trade-off between memory and time resources since it requires more memory to save the intermediate outputs but requires fewer point evaluations to evaluate epigenetically mutated individuals. The trade-off is favorable in our implementation because processor resources are much more limited than memory resources. Similar partial evaluation strategies have been proposed, e.g., in Langdon (2012).

5.3. Problems

The methods are first compared on a set of coupled, two-state nonlinear ODEs adapted from Strogatz (2014) and proposed in Schmidt (2011). Second, they are tested for scalability against a suite of hundreds of randomly generated ODE problems with varying complexity and dimensionality. Finally, ELGP is applied to the identification of nonlinear dynamics of three real-world processes, including a pump-fed system of cascaded water tanks, an industrial chemical distillation tower, and an industrial wind turbine.

5.3.1. Textbook ODE problems

The textbook ODE problems represent seven two-state, nonlinear systems from the fields of biology, electrical engineering, physics, ecology, and fluid dynamics. For brevity, the form of the models is shown alongside identification results in Table 3. In accordance with Schmidt (2011), each system is simulated for 10 seconds from 4 different initial conditions chosen randomly within stable basins of attraction, giving a total of 400 data points for training. The settings for each problem are summarized in Table 1. In order to give a measure of nonlinearity and/or difficulty of these identification problems, we also use multiple linear regression (LR) to estimate models for these systems. The LR models are estimated as a weighted sum of the states (in this case the systems have no external inputs), i.e. $\hat{y} = \hat{\beta}^T\mathbf{x}$, where $\hat{\beta}$ is the least-squares solution minimizing the sum of the squared prediction error: $\sum_{k=1}^{N}(y(t_k) - \hat{\beta}^T\mathbf{x}(t_k))^2$.

5.3.2. ODE suite

In order to test the scalability of the methods, random ODE systems were generated of varying size (nodes) and dimensionality (number of variables). This approach to scalability testing is used in order to remove problem selection bias and to quantify the methods' performance with different target complexity (Schmidt and Lipson, 2007; Cornforth and Lipson, 2013). The dynamic systems were generated in the following fashion. First, differential equations were randomly generated using the same equation generation technique that initializes populations of equations. Second, the equations were simulated as first-order differential equations (using Runge-Kutta 4) according to a random set of initial conditions from the range [−5,5]. The outputs of these simulation runs were included in the training set. The validation set was subsequently generated by simulating the equations with initial conditions randomly selected from the range [−10,10]. Equations that produced invalid outputs were discarded. Finally, the valid equations were simplified symbolically in MATLAB in order to determine their most succinct representation, and binned by number of nodes and dimensions. The result of the entire process was 640 unique ODE problems of 3 to 33 nodes and 1 to 8 variables that were subjected to 10 trials of identification, for a total of 6400 trials per GP treatment. The ODE suite settings are summarized in Table 2.

5.3.3. Real-world applications

We consider three real-world applications of ELGP. The first is a benchmark cascaded tanks system consisting of three measured

Table 2
ODE suite problem settings.

ODE suite settings	
Number of nodes	3–33
Dimensions	1–8
Models per setting	5
Trials per model	10
Total models	640
Total trials	6400
Function set	{+, −, *,/, sin, cos, exp, log}

variables. The second two come from complex industrial processes with 25 (chemical tower) and 8 (wind turbine) measured variables. All three processes are known to exhibit nonlinear dynamics.

In order to analyze ELGP's performance in the context of other nonlinear modeling approaches, we compare the quality of the ELGP's solutions to these problems with solutions from standard linear and nonlinear empirical modeling approaches. The models were developed using ARX and two NARX approaches with different nonlinear transformations: wavelet networks (NARX-W) and feed-forward neural networks (NARX-NN). We used the MATLAB System Identification toolbox (Ljung, 2007) to generate these models using default settings. For the wavelet network, the set of nonlinear regressors was computed using a radial wavelet expansion with an automatically determined number of terms. For the NARX-NN, a feedforward network with 10 hidden neurons was constructed and trained with back-propagation learning using the Levenberg-Marquardt algorithm. In order to provide uniformity among the predicted outputs of these models and that of ELGP's solutions, the ELGP models were simulated on the test sets, such that prediction error (Eq. (3)) was defined in terms of the simulated output $\hat{y}(t_k)$ rather than the state derivatives.

Benchmark system. In order to study the performance of ELGP on a real-world benchmark problem, we performed identification based on a set of observations collected from two cascaded tanks fed by a water pump (Wigren and Schoukens, 2013). Using the Bernoulli principle and mass conservation, this system can be represented by the following nonlinear equations:

$$\begin{aligned}\dot{h}_1 &= -\theta_1\sqrt{h_1} + \theta_2 u(t) + w_1(t)\\ \dot{h}_2 &= \theta_1\sqrt{h_1} - \theta_3\sqrt{h_2} + w_2(t)\end{aligned} \tag{12}$$

$$\begin{aligned}y_1 &= h_1(t) + e_1(t)\\ y_2 &= h_2(t) + e_2(t)\end{aligned} \tag{13}$$

where $\theta_1 = -\frac{a_1\sqrt{2g}}{A_1}$, and $\theta_2 = -\frac{k}{A_1}$, $\theta_3 = \frac{a_2\sqrt{2g}}{A_2}$. States h_1 and h_2 represent the water levels in the upper and lower tanks, respectively; a_1 and a_2 are the outlet areas; A_1 and A_2 are the horizontal cross sections of the tanks; g is the gravitational constant; k is the pump voltage to flow conversion constant; $w_1(t)$ and $w_2(t)$ are system noise; and $e_1(t)$ and $e_2(t)$ are measurement noise.

The data set comprises 2500 samples, acquired at a sampling period of 5 seconds. We divided this set 50/50 for training and testing. These data have been proposed for benchmarking nonlinear system identification approaches (Wigren, 2006) and are freely available (Wigren, 2010). For ELGP, we use `EHC5` with the settings of Table 1 and an increased function set {+, −, *,/, sin, cos, exp, log, $\sqrt{()}$}. For ARX and NARX, identification of these models was performed as a first-order function of the input, i.e. with the regressors $y_1(t_k-1)$, $y_2(t_k-1)$, and $u(t_k)$.

Industrial processes. In addition to finding exact solutions to known systems, ELGP should be able to identify reliable models of real-world systems for which no true model is known. To this end, two complex industrial problems are also considered in this work. The first is the Tower problem,[2] consisting of a set of 15-min averaged time series data taken from a chemical distillation tower, totaling 3135 samples. The goal is to predict the propylene concentration at the top of the tower from 25 measured variables. The second problem, referred to as the Wind problem, features data collected from the Controls and Advanced Research Turbine, a 600 kW wind turbine operated by the National Wind Technology Center (Fleming et al., 2011). The data set consists of 6000 time-series measurements of wind speed, control actions, and acceleration measurements that are used to predict the bending moment measured at the base of the wind turbine.

For these two problems, solutions are formulated as fifth-order discrete-time dynamic models of the form $\hat{M}(\hat{y}(t_k-1)\ldots\hat{y}(t_k-5), \zeta(t_k)\ldots\zeta(t_k-5), \hat{\boldsymbol{\Theta}})$ in terms of model outputs $\hat{y}$, measured variables ζ, and model parameters $\hat{\boldsymbol{\Theta}}$. This formulation was used with the ELGP, ARX and NARX-NN methods.

6. Results and discussion

We first present results obtained on the textbook ODE problems in Section 6.1. Comparisons include the number of exact solutions, the fitness of the best solutions, and complexity of the best models found by each method. Next we analyze in Section 6.2 the ODE suite results according to fitness as a function of point evaluations in training and testing over the entire suite. To give a sense of the scalability of the methods, we group the results by target complexity and compare the number of solutions found and the computational effort spent to reach these solutions. We then compare the ELGP solution with a few black-box models obtained from the experimental data of a real world benchmark identification problem in Section 6.3 and two industrial problems in Section 6.4. We end our analysis with a detailed look at population diversity of the Bacterial Respiration problem, which provides insight into the mechanics of the variants of ELGP (Ep1M, EHC1, EHC5, EHC10), particularly the EHC methods (EHC1, EHC5, EHC10), that lead to the better solutions obtained on many of the studied problems.

6.1. Textbook ODE problems

As a preliminary evaluation of the intelligibility of ELGP's solutions, the most frequent solutions to the textbook ODE problems that were found using Base and the ELGP variant EHC5 are compared against the target model forms in Table 3. Particularly noteworthy are the differences observed between the Base and EHC5 solutions for the Bacterial Respiration (1 and 2), Bar Magnets (1), Predator Prey (1 and 2), and van der Pol oscillator (1) identifications. In each of these cases, EHC5 more often identifies the exact solution, or at least an approximation of it that is less complex than that inferred by Base. In some cases, e.g. Bacterial Respiration 1 and van der Pol 1, the approximate solutions from EHC5 have most of the terms of the target correct, and thus form a sensible approximation of the true system. Note that the Base solution to Predator Prey 2, i.e.,

$$\dot{y} = (y + 0.01059)\cdot\left(\frac{x}{x+1.004} - 0.07488\cdot y\right)$$

could be made more correct through the change $(y + 0.01059) \rightarrow y$. This type of topological model change is easily reached via epigenetic transformations, and, as is shown in

[2] http://symbolicregression.com/?q=towerProblem.

Table 3
The textbook ODE problems (left). The models generated by Base and ELGP variant EHC5 are shown on the right.

System	Target	Base Most Frequent Solution	EHC5 Most Frequent Solution
Bacterial Respiration	$\dot{x} = 20 - x - \frac{x \cdot y}{1+0.5 \cdot x^2}$	$\dot{x} = -(7.006 \cdot (x + 0.4722 \cdot y - 31.11))/y$	$\dot{x} = 20.12 - 1.009 \cdot x - \frac{1.979 \cdot y}{x}$
	$\dot{y} = 10 - \frac{x \cdot y}{1+0.5 \cdot x^2}$	$\dot{y} = 9.994 - 0.5669 \cdot y \cdot sin(3.526/x)$	$\dot{y} = 10.0 - \frac{2.0 \cdot x \cdot y}{2.001+x^2}$
Bar Magnets	$\dot{\theta} = 0.5 \cdot \sin(\theta - \phi) - \sin(\theta)$	$\dot{\theta} = sin(sin(cos(cos(\theta) - \phi) + sin(\theta) - 3.65))$	$\dot{\theta} = 0.5 \cdot sin(\theta - \phi) - sin(\theta)$
	$\dot{\phi} = 0.5 \cdot \sin(\phi - \theta) - \sin(\phi)$	$\dot{\phi} = -0.5 \cdot sin(\theta - \phi) - sin(\phi)$	$\dot{\phi} = -0.5 \cdot sin(\theta - \phi) - sin(\phi)$
Glider	$\dot{v} = -0.05 \cdot v^2 - sin(\theta)$	$\dot{v} = -0.05 \cdot v^2 - sin(\theta)$	$\dot{v} = -0.05 \cdot v^2 - sin(\theta)$
	$\dot{\theta} = v - \cos(\theta)/v$	$\dot{\theta} = v - (cos(\theta))/v$	$\dot{\theta} = v - (cos(\theta))/v$
Lotka-Volterra interspecies dynamics	$\dot{x} = 3 \cdot x - 2 \cdot x \cdot y - x^2$	$\dot{x} = 3.0 \cdot x - 2.0 \cdot x \cdot y - x^2$	$\dot{x} = 3.0 \cdot x - 2.0 \cdot x \cdot y - x^2$
	$\dot{y} = 2 \cdot y - x \cdot y - y^2$	$\dot{y} = 2.0 \cdot y - x \cdot y - y^2$	$\dot{y} = 2.0 \cdot y - x \cdot y - y^2$
Predator Prey	$\dot{x} = x \cdot \left(4 - x - \frac{y}{1+x}\right)$	$\dot{y} = -0.5674 \cdot x \cdot (y + x \cdot cos(\frac{0.7896 \cdot y}{x+cos(0.1632 \cdot x)}) - 6.119)$	$\dot{y} = 0.3516 \cdot x \cdot \left(8.122 - \frac{x}{cos(x/y)} - y\right)$
	$\dot{y} = y \cdot \left(\frac{x}{1+x} - 0.075 \cdot y\right)$	$\dot{y} = (y + 0.01059) \cdot \left(\frac{x}{x+1.004} - 0.07488 \cdot y\right)$	$\dot{y} = y \cdot \left(\frac{x}{1.0+x} - 0.075 \cdot y\right)$
Shear Flow	$\dot{\theta} = \cot(\phi) \cdot cos(\theta)$	$\dot{\theta} = \cot(\phi) \cdot cos(\theta)$	$\dot{\theta} = \cot(\phi) \cdot cos(\theta)$
	$\dot{\phi} = \left(\cos^2(\phi) + 0.1 \cdot \sin^2(\phi)\right) \cdot sin(\theta)$	$\dot{\phi} = 0.45 \cdot sin(\theta) \cdot (cos(2.0 \cdot \phi) + 1.222)$	$\dot{\phi} = 2.076 \cdot sin\,(cos(sin(\phi)) - 0.4918) \cdot sin(\theta)$
van der Pol oscillator	$\dot{x} = 10 \cdot \left(y - \frac{1}{3} \cdot (x^3 - x)\right)$	$\dot{x} = (y + 10.41) \cdot (2.232 \cdot sin(x) - 1.879 \cdot x + sin(y))$	$\dot{x} = 10.0 \cdot \left(y - 0.333 \cdot (x^3 - x)\right)$
	$\dot{y} = -\frac{1}{10} \cdot x$	$\dot{y} = -0.1 \cdot x$	$\dot{y} = -0.1 \cdot x$

Table 3, EHC5 more frequently identifies the underlying target model for this problem.

A central goal of ELGP is to bolster the performance of GP in system identification through local topological search. To this end, the number of solutions, median best fitness and average equation size (number of active nodes) for the different methods are summarized in Tables 4 and 5. Pairwise statistical comparisons are given for each result. Note that three of the identification tasks (Glider 2, Shear Flow 1, and van der Pol 2) are exactly identified 100% of the time by every GP treatment, suggesting that they are easy for GP to solve. For the eleven other problems, the results show that the training and test fitnesses and solution counts are improved by EHC. For example, on each of these eleven problems, the ELGP variant EHC10 finds significantly ($p < 0.05$) more exact solutions and produces models with better training and test fitnesses than Base, Ep0 or Ep1M, as indicated by highlighting, bold text, and * in the tables. In terms of fitness, EHC5 provides a significant improvement relative to Base on eight out of eleven and EHC1 performs better on five out of eleven problems, thus suggesting that results improve with more iterations of EHC. Among the eleven more difficult problems, all of the EHC methods perform significantly better than Ep0 or Ep1M, in terms of fitness as well as exact solutions. Overall the results of Ep0 or Ep1M show a marginal to negative difference in estimation capacity compared to Base. The guided search provided by EHC, therefore, is key to the observed improvements.

The results indicate that every GP method produces better models than linear regression (LR) for these problems (with the exception of the linear second state of the van der Pol problem), which is to be expected given the known nonlinear nature of this set of problems. However, LR has the advantage of quick training times, with median convergence times on the order of 0.01 s for these problems, as shown in the last column of Tables 4 and 5. The GP methods converge to the minimum fitness model in approximately 1–100 s, depending on the difficulty of the problem. It is worth noting that the EHC methods do not increase the computation time compared to Base, and occasionally decrease it, which can be attributed to the lower proportion of optimization spent conducting GP generations and the higher proportion spent in EHC. The majority of ELGP computation times are not significantly different from Base. These convergence times suggest that the proposed identification methods may be suitable for certain online applications, depending on the time window constraints between model updates.

In addition to improving predictive ability, a motivation for the ELGP design is the delivery of concise solutions. This property is evident from the average program sizes in Tables 4 and 5. To further evaluate this aspect of the results, in Fig. 4, the best solutions of the 100 trials are evaluated in terms of *Solution Bloat*, defined as the difference in complexity (Eq. (8)) between the GP solution and the target. These results show that the ELGP variants all produce solutions that are more succinct than those achieved by Base. Among ELGP variants, Ep0 or Ep1M produce the most succinct models. The EHC methods also produce more succinct models with less solution bloat than Base; however, it is observed that the hill climbing aspect of EHC leads to slightly larger models than Ep0 or Ep1M. Nevertheless, in addition to producing more succinct models than Base, the EHC methods find exact solutions more often for most problems, as shown in Tables 4 and 5.

6.2. ODE suite

To evaluate the ability of ELGP to scale to problems of increasing complexity, we evaluate the performance of the treatments Base, Ep0, Ep1M, EHC1 and EHC5 on a suite of 640 randomly generated target systems. The best fitness on training and test sets for the entire ODE suite is shown in Figs. 5 and 6 as a function of point evaluations. The results indicate the better fitness minimization properties of the ELGP variants EHC1 and EHC5. The number of solutions found, and the computational effort to reach those solutions, are plotted in Figs. 7 and 8, respectively. The results are broken into groups based on the number of arguments and operands (i.e. nodes) in the solution equation, for example 7, 9, 11, and so on. Fig. 7 demonstrates the ability of the ELGP variants, especially EHC5, to find more solutions than Base or Ep0, and the difference in performance grows as the number of nodes in the solution increases, although the overall number of solutions

Table 4
Comparison of best-of-run results for the ODE benchmark problems. Statistical significance ($p < 0.05$) is denoted as follows: °: better than MR; (.): better than Base; **bold**: better than Ep0; *: better than Ep1M; †: better than EHC1; ‡: better than EHC5. Exact solution p-values are based on pairwise chi-squared tests with Holm correction. Fitness and bloat p-values are based on pairwise Wilcoxon rank-sum tests.

Problem	Method	Exact Solutions (%)	Median Best Fitness	Median Program Size	Median Convergence Time (s)
Bacterial Respiration 1	MR	0	0.21196	n/a	†‡***0.01**
	Base	0	°**0.14519**	15.61	38.77
	Ep0	0	°0.16068	†‡ 10.33	35.35
	Ep1M	0	°0.15657	†‡ 10.42	50.42
	EHC1	0	*°**0.14160**	‡ 11.91	46.14
	EHC5	0	*°**0.13907**	12.52	29.03
	EHC10	0	†*°**0.13245**	13.00	52.35
Bacterial Respiration 2	MR	0	0.21273	n/a	†‡***0.01**
	Base	0	°0.00785	15.54	74.09
	Ep0	0	°0.00794	†‡ 10.29	31.31
	Ep1M	0	°0.00957	†‡ 10.29	71.70
	EHC1	0	*°**0.00741**	‡ 11.58	94.61
	EHC5	0	†*°**0.00634**	12.19	77.55
	EHC10	0	†*°**0.00617**	12.19	75.22
Bar Magnet 1	MR	0	0.04057	n/a	†‡***0.01**
	Base	2	†*°**0.02351**	14.84	87.13
	Ep0	1	°0.02866	†‡ 10.32	74.07
	Ep1M	2	°0.03155	†‡ 10.35	83.61
	EHC1	7	*°**0.02733**	‡ 11.31	48.95
	EHC5	7	†*°**0.02242**	12.24	***39.49**
	EHC10	7	†*°**0.02103**	12.64	***50.79**
Bar Magnet 2	MR	0	1.19817	n/a	‡0.01
	Base	°18	*°**0.01254**	15.51	68.79
	Ep0	°16	°0.01502	†‡ 10.77	61.54
	Ep1M	°16	°0.01729	†‡ 10.72	81.35
	EHC1	*°**41**	*°**0.00005**	11.59	68.86
	EHC5	*°**45**	*°**0.00000**	12.00	93.17
	EHC10	*°**38**	*°**0.00001**	12.34	‡80.07
Glider 1	MR	0	4.52904	n/a	0.01
	Base	°75	°0.00000	7.22	82.65
	Ep0	°77	°0.00000	5.07	91.69
	Ep1M	°76	°0.00000	4.25	64.20
	EHC1	*°**99**	*°**0.00000**	* **3.03**	53.92
	EHC5	*°**100**	*°**0.00000**	* **3.26**	73.33
	EHC10	*°**100**	*°**0.00000**	**3.53**	81.55
Glider 2	MR	0	0.53583	n/a	0.01
	Base	100	°0.00000	2.76	19.89
	Ep0	100	°0.00000	‡ 1.71	7.61
	Ep1M	100	°0.00000	‡ 1.62	6.11
	EHC1	100	°0.00000	‡ 1.49	4.99
	EHC5	100	°0.00000	2.46	6.37
	EHC10	100	°0.00000	‡ 1.57	4.14
Lotka-Volterra 1	MR	0	0.94071	n/a	†‡0.01
	Base	7	*°**0.04218**	16.30	48.83
	Ep0	°10	°0.06897	†‡ 11.07	83.16
	Ep1M	8	°0.06805	†‡ 11.24	59.40
	EHC1	°13	*°**0.00050**	‡ 12.71	82.81
	EHC5	*°**30**	†*°**0.00003**	13.20	73.86
	EHC10	†*°**43**	†‡*°**0.00000**	13.18	67.98

decreases with more complex targets. In addition, EHC1 and EHC5 tend to find solutions with less computational effort than the other methods, as demonstrated in Fig. 8, where we plot the number of point evaluations to termination for the trials in which exact solutions were found. The results also suggest that the computational effort improvement afforded by EHC increases with the complexity of the problem.

6.3. Real-world benchmark system

In this section, EHC5 and several standard modeling approaches are applied to the identification of a benchmark system of cascaded tanks fed by a pump (Wigren, 2010). Since the true parameters of the system are not provided in Wigren (2010), we first estimate the parameters of the model in Eq. (13) by linear regression, which yields

$$\begin{aligned}\hat{\dot{y}}_1 &= -0.0122\sqrt{\hat{y}_1} + 0.0188\,u(t)\\ \hat{\dot{y}}_2 &= -0.0481\sqrt{\hat{y}_1} + 0.0452\sqrt{\hat{y}_2}\end{aligned} \tag{14}$$

In comparison, EHC5 applied to the measured data yields

$$\begin{aligned}\hat{\dot{y}}_1 &= -\hat{\theta}_1\sqrt{\hat{y}_1} + \hat{\theta}_1\sqrt{\hat{y}_2}\\ \hat{\dot{y}}_2 &= -\hat{\theta}_2\sqrt{\hat{y}_1} + \hat{\theta}_2(u(t) + \hat{\theta}_3)\end{aligned} \tag{15}$$

where $\hat{\theta}_1 = 0.0905$, $\hat{\theta}_2 = 0.0302$, and $\hat{\theta}_3 = 0.6845$. The model in Eq. (15) is an interesting permutation of the theoretical model (Eq. (14)) in that it correctly identifies the square root nonlinearities of the water levels and is as succinct as the theoretical model. However Eq. (15) incorrectly associates the pump input $u(t)$ and top tank level $y_1(t)$ with the second state derivative, and uses the theoretical form of $\hat{\dot{y}}_2(t)$ in Eq. (14) for the first state derivative $\hat{\dot{y}}_1$

Table 5
Comparison of best-of-run results for the ODE benchmark problems. Statistical significance ($p < 0.05$) is denoted as follows: $^\diamond$: better than MR; $(\cdot)$: better than Base; **bold**: better than Ep0; *: better than Ep1M; †: better than EHC1; ‡: better than EHC5. Exact solution p-values are based on pairwise chi-squared tests with Holm correction. Fitness and bloat p-values are based on pairwise Wilcoxon rank-sum tests.

Problem	Method	Exact Solutions (%)	Median Best Fitness	Median Program Size	Median Convergence Time (s)
Lotka-Volterra 2	MR	0	0.19556	n/a	0.01
	Base	$^\diamond$53	$^\diamond$0.00000	14.51	89.27
	Ep0	$^\diamond$73	$^\diamond$**0.00000**	6.33	59.00
	Ep1M	$^\diamond$67	$^\diamond$**0.00000**	7.04	77.29
	EHC1	$^{*\diamond}$**90**	$^{*\diamond}$**0.00000**	* 4.77	84.41
	EHC5	$^{*\diamond}$**97**	$^{*\diamond}$**0.00000**	* **4.31**	68.87
	EHC10	$^{*\diamond}$**99**	$^{*\diamond}$**0.00000**	†* **3.19**	76.08
Predator Prey 1	MR	0	2.42447	n/a	††***0.01**
	Base	0	$^{*\diamond}$**0.17383**	16.28	76.36
	Ep0	0	$^\diamond$0.19041	†‡ 10.89	66.30
	Ep1M	0	$^\diamond$0.19159	†‡ 10.89	84.51
	EHC1	0	$^{*\diamond}$**0.18286**	‡ 12.61	62.70
	EHC5	0	†$^{*\diamond}$**0.16950**	13.37	71.86
	EHC10	0	†$^{*\diamond}$**0.16023**	13.54	79.33
Predator Prey 2	MR	0	0.31338	n/a	††***0.01**
	Base	0	†$^{*\diamond}$**0.20065**	16.72	86.67
	Ep0	0	$^\diamond$0.23531	†‡ 11.41	90.11
	Ep1M	0	$^\diamond$0.23249	†‡ 11.38	††29.60
	EHC1	0	$^{*\diamond}$**0.20694**	‡ 12.86	84.19
	EHC5	1	†$^{*\diamond}$**0.19156**	13.69	85.89
	EHC10	0	†$^{*\diamond}$**0.19136**	13.78	52.34
Shear Flow 1	MR	0	3.33019	n/a	0.01
	Base	$^\diamond$100	$^\diamond$0.00000	††‡1.42	9.50
	Ep0	$^\diamond$100	$^\diamond$0.00000	††‡1.35	4.00
	Ep1M	$^\diamond$100	$^\diamond$0.00000	††‡1.16	4.43
	EHC1	$^\diamond$100	$^\diamond$0.00000	‡1.83	2.70
	EHC5	$^\diamond$100	$^\diamond$0.00000	2.17	2.00
	EHC10	$^\diamond$100	$^\diamond$0.00000	††‡1.11	3.43
Shear Flow 2	MR	0	0.56694	n/a	††***0.01**
	Base	0	$^\diamond$0.00100	16.68	***32.31**
	Ep0	1	$^\diamond$0.00111	†‡ 10.94	104.38
	Ep1M	0	$^\diamond$0.00099	†‡ 11.00	106.03
	EHC1	0	$^{*\diamond}$**0.00092**	‡ 11.96	76.78
	EHC5	1	$^{*\diamond}$**0.00093**	12.32	83.49
	EHC10	0	$^{*\diamond}$**0.00040**	12.41	*90.77
van der Pol 1	MR	0	8.90324	n/a	††***0.01**
	Base	0	$^{*\diamond}$0.23003	18.81	87.86
	Ep0	0	$^\diamond$0.24255	†‡ 13.87	82.28
	Ep1M	0	$^\diamond$0.27929	†‡ 13.95	78.77
	EHC1	1	$^{*\diamond}$**0.20830**	14.84	85.48
	EHC5	3	†$^{*\diamond}$**0.10080**	14.95	69.16
	EHC10	0	†$^{*\diamond}$**0.10159**	15.19	66.99
van der Pol 2	MR	100	0.00000	n/a	0.01
	Base	100	0.00000	†*1.31	4.21
	Ep0	100	0.00000	2.02	1.66
	Ep1M	100	0.00000	**1.51**	2.35
	EHC1	100	0.00000	**1.54**	2.28
	EHC5	100	0.00000	†***1.34**	3.34
	EHC10	100	0.00000	2.30	4.78

in Eq. (15). In comparison to the theoretical model, however, the ELGP solution produces much better predictions, as illustrated by the time series comparison in Fig. 9.

For this system, we compare the test set accuracy of the EHC5 model to some state-of-the-art black-box models in Table 6, in terms of the number of parameters in the resultant model, and the mean square error (MSE) and R^2 (Eq. (5)) of simulated outputs on the test set. The most accurate model predictions are generated by NARX-NN, both in terms of MSE and R^2, followed by ELGP, which ties NARX-NN in prediction correlation for $y_2(t)$. ARX is the next most accurate on average, followed by the theoretical model and NARX-W. In comparison to the predictions of $\hat{y}_1$ and $\hat{y}_2$ from the ARX model, the ELGP model has a 42% and 18% lower mean absolute error, and 4% and 1% better R^2 value. The inaccuracy of NARX-W is surprising given its complexity (64 parameters), and suggests a mismatch between the assumed nonlinearities of the approach and those present in the measured system. The NARX-NN model's excellent predictions come at the expense of complexity: the model consists of two networks with 10 hidden neurons each, totaling 60 learned parameters per model. In this sense the ELGP solution is quite reassuring because it achieves reasonable accuracy in prediction with half the parameters of the ARX model and 57 less parameters than the NARX-NN model.

The difficulty in estimating the correct model form for this system (Eq. (14)) may stem from the similarity of the measured outputs $y_1(t)$, $y_2(t)$, as is shown in Fig. 9, as well as the effects of system noise and measurement error ($e_1(t)$, $e_1(t)$, $w_1(t)$, $w_2(t)$). It is especially clear from Fig. 9 that the measured $y_2(t)$ deviates substantially from its theoretical behavior, making the theoretical model difficult to infer. More fundamentally, the identification difficulty could stem from an intrinsic difficulty for ELGP in handling processes of this form subject to the measured input conditions. To determine whether or not the intrinsic difficulty of this system affects ELGP's results, we simulate the theoretical system given in Eq. (14) using the measured

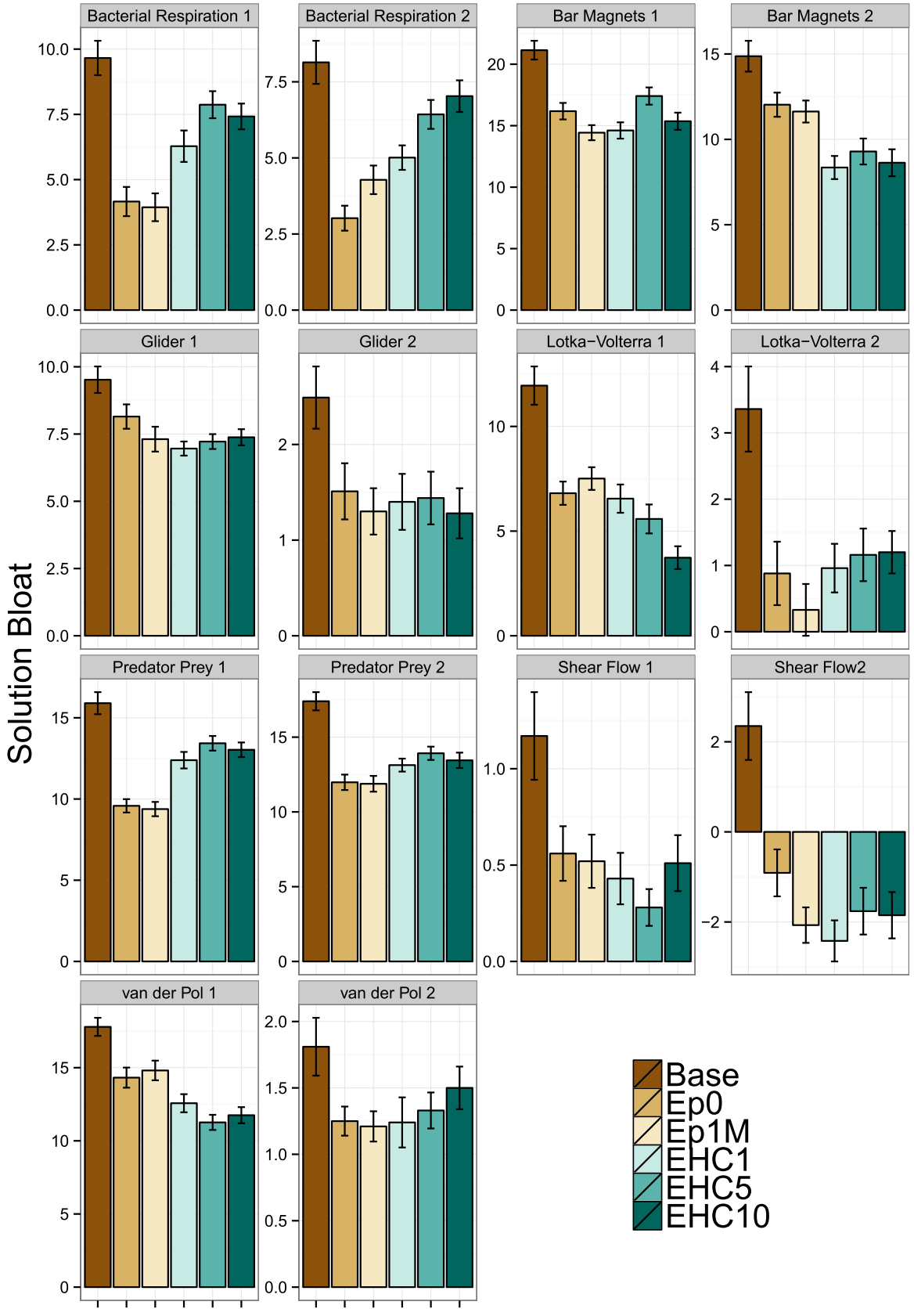

Fig. 4. Comparison of solution bloat for the textbook ODE problems.

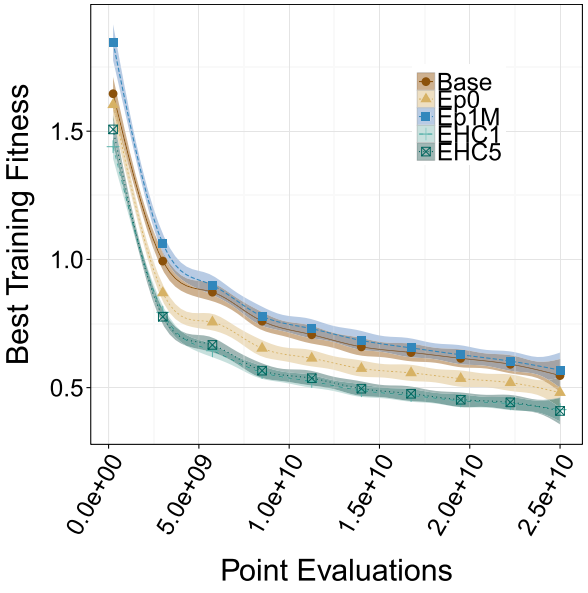

Fig. 5. Fitness on the training set for the ODE suite.

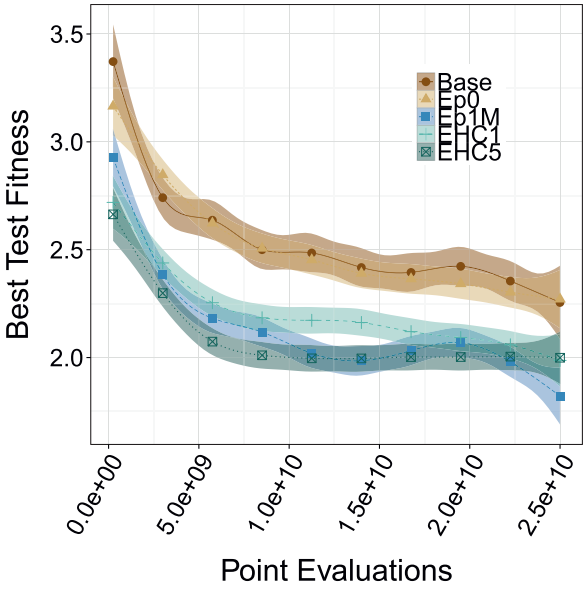

Fig. 6. Fitness on the test set for the ODE suite.

input and use the resulting outputs $(\hat{y}_1(t), \hat{y}_2(t), u(t))$ to train models using EHC5. In this case, EHC5 renders a nearly perfect model of the theoretical system:

$$\begin{aligned}\hat{\dot{y}}_1 &= -\hat{\theta}_1\sqrt{(\hat{\theta}_2\hat{y}_1)} + \hat{\theta}_1 u(t)\\ \hat{\dot{y}}_2 &= -\hat{\theta}_3\sqrt{\hat{y}_1} + \hat{\theta}_3\sqrt{(\hat{y}_2/\hat{\theta}_4)}\end{aligned} \tag{16}$$

with $\hat{\theta}_1 = 0.0188$, $\hat{\theta}_2 = 0.4200$, $\hat{\theta}_3 = 0.0452$, and $\hat{\theta}_4 = 0.8830$. Both states in Eq. (16) have an MSE $< 10^{-6}$ and $R^2 = 1$. Thus the mismatch between the ELGP model form and the assumed process physics appears to arise from system noise and measurement error.

6.4. Industrial processes

The GP variants were used to identify models for two industrial processes: the Tower problem and Wind problem. We compare the fitness of the different GP methods as a function of computational effort (point evaluations) for the Tower and Wind problems in Fig. 10.

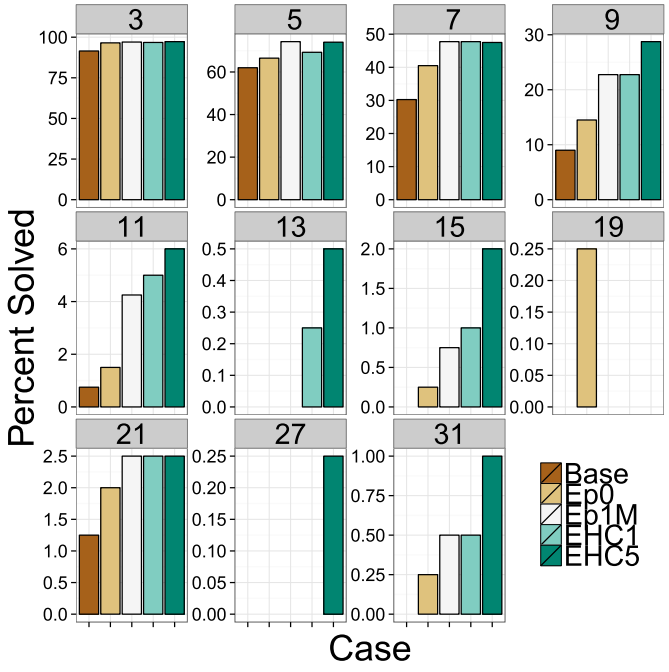

Fig. 7. Percent of solutions found for the ODE suite. Results are grouped based on the number of nodes in the target equation (labeled at the top).

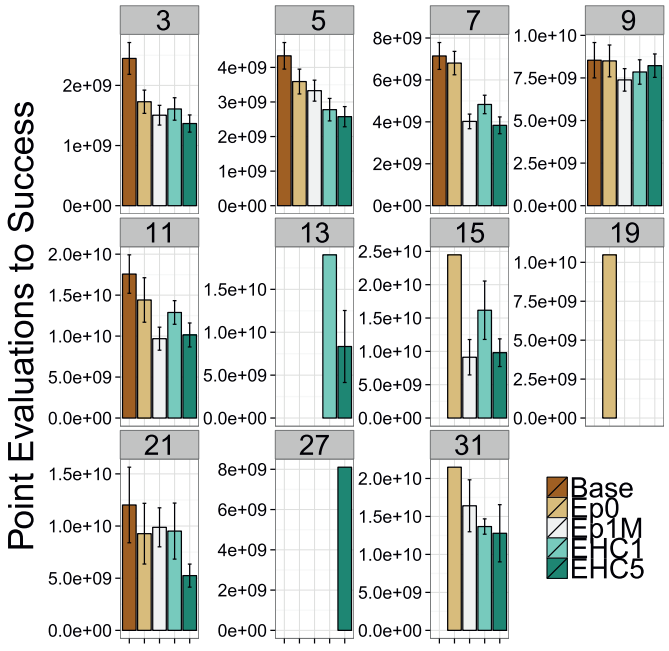

Fig. 8. Point evaluations to success for the ODE suite. Results are grouped based on the number of nodes in the target equation (labeled at the top).

For both problems, EHC1 converges most quickly to the minimum fitness found for the problems. For the Tower problem, we observe that Ep1M performs well on the test set, and as suggested in previous work (La Cava et al., 2015), this may be due to the compact models produced by this method for this problem, since smaller models tend to generalize better than larger models. On the Wind problem, EHC1 and EHC5 converge the quickest and produce the best fitness on the test set. Ep0 and EHC10 tend to perform worse than EHC1, suggesting that EHC10 is overly greedy for these problems. Overall, the ELGP variants perform significantly better than Base on both problems, converging approximately twice as fast as Base and producing better fit models.

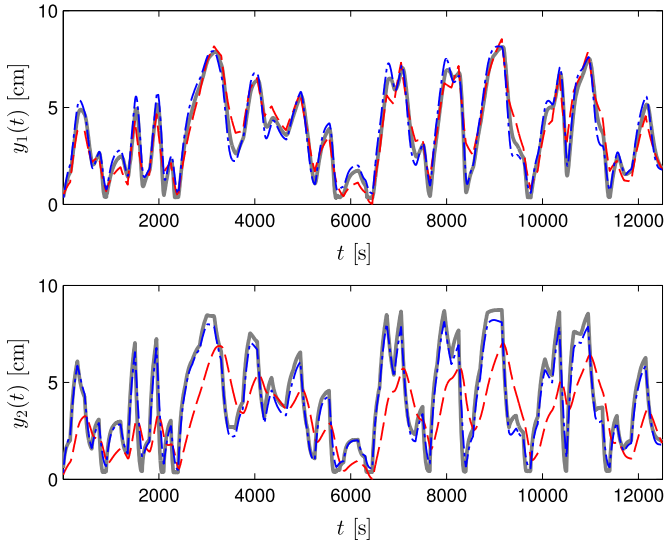

Fig. 9. Comparison of outputs for the cascaded tanks problem, including measurement data (solid gray), the theoretical model (Eq. (14), dashed red), and the ELGP model (Eq. (15), dot-dashed blue). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 6
Mean square error (MSE) and R^2 values of model predictions on the test sets for the cascaded tanks problem using several approaches.

Model	Parameters	MSE (test)		R^2 (test)	
		$y_1(t)$	$y_2(t)$	$y_1(t)$	$y_2(t)$
Theoretical	3	0.463	5.343	0.907	0.313
ARX	6	0.432	0.350	0.919	0.961
ELGP (EHC5)	3	0.249	0.288	0.953	0.974
NARX-W	64	2.960	6.165	0.807	0.748
NARX-NN	60	0.120	0.182	0.977	0.974

We compare the best ELGP models to models identified using ARX and NARX-NN in Table 7. For the Tower problem, the ELGP model is a notable improvement over the linear model (note the 279% higher R^2 value), but is slightly less accurate than the model generated using NARX-NN (4% lower R^2). However it is important to note that the high number of variables in this problem produce a neural network with 250 parameters, whereas the ELGP model contains only 7. For the Wind problem, ELGP produces the most accurate model among the tested methods. The MAE of the models

Table 7
Mean square error (MSE) and R^2 values of model predictions on the test sets for the Tower and Wind problem using several approaches.

Problem	Method	Parameters	MAE (test)	R^2 (test)
Tower	ARX	130	55.190	0.028
	ELGP	7	30.753	0.811
	NARX-NN	250	22.439	0.846
Wind	ARX	35	0.026	0.642
	ELGP	5	0.025	0.773
	NARX-NN	60	0.026	0.727

generated by each method are similar; however, the correlation of the ELGP model to test data is 20% better than the ARX model, and 6% better than the NARX-NN model. The ELGP model also contains significantly fewer parameters than the ARX or NARX-NN models. Hence the conciseness of the ELGP models is an encouraging trade-off for the accuracy they exhibit.

6.5. Population diversity

Results on the textbook ODEs and the ODE suite suggest that the EHC methods improve GP performance significantly. We hypothesize that this improvement is created by preserving sections of the genome from fitness pressure and allowing them to drift genetically, thus providing an avenue for introduction of diversity and continued progress towards the solution. To test this hypothesis, the syntactic and semantic similarity of models in the population can be examined in detail to determine whether this phenomenon of preserved diversity is evident. We define syntactic similarity as the homology H of a population using a Levenshtein distance comparison of S randomly sampled pairs of individuals $(\mathbf{i}_j, \mathbf{i}_m|_L)$ normalized by the length $(|\cdot|)$ of the longer individual, as

$$H = 1 - \frac{1}{S}\sum_{n=1}^{S}\frac{|\mathbf{i}_j, \mathbf{i}_m|_L}{\max(|\mathbf{i}_j|, |\mathbf{i}_m|)} \tag{17}$$

We sample H each generation for both the active and inactive portions of genomes with $S=200$. In addition, we define the semantic; i.e., behavioral, similarity of the population, referred to as *Similar Behavior*, as the fraction of identical output vectors among models in the population. This allows us to compare what is happening genetically and epigenetically at the program level (syntax) to the behavior of the resultant models (semantics).

In general we find that *silenced* genotypes have lower

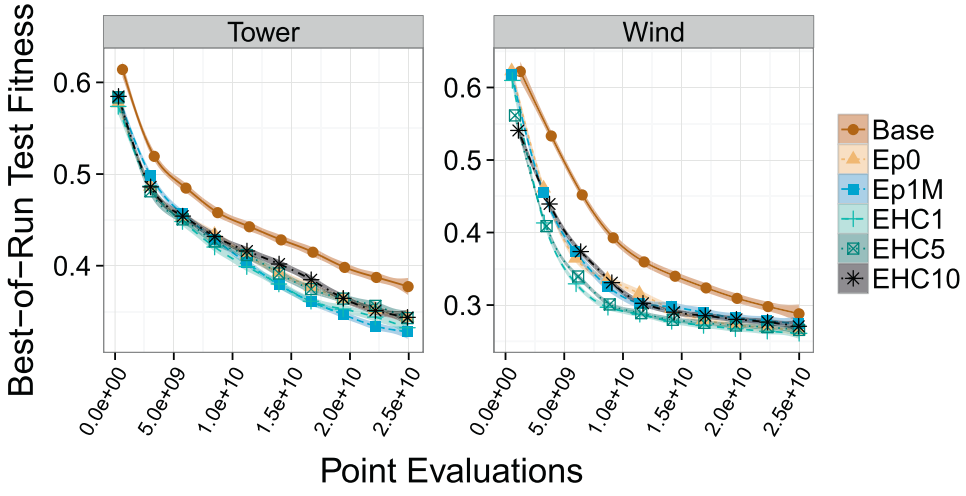

Fig. 10. Fitness on the test set for the industrial problems. Filled area indicates the 95% confidence interval.

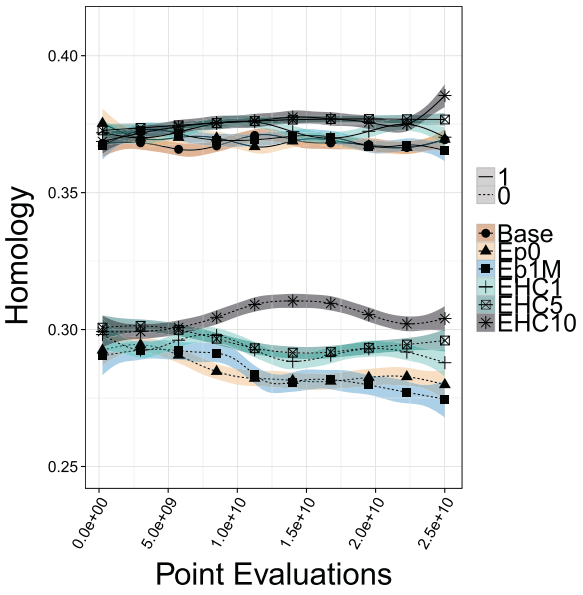

Fig. 11. Homology among active and inactive genomes for the Bacterial Respiration 2 problem.

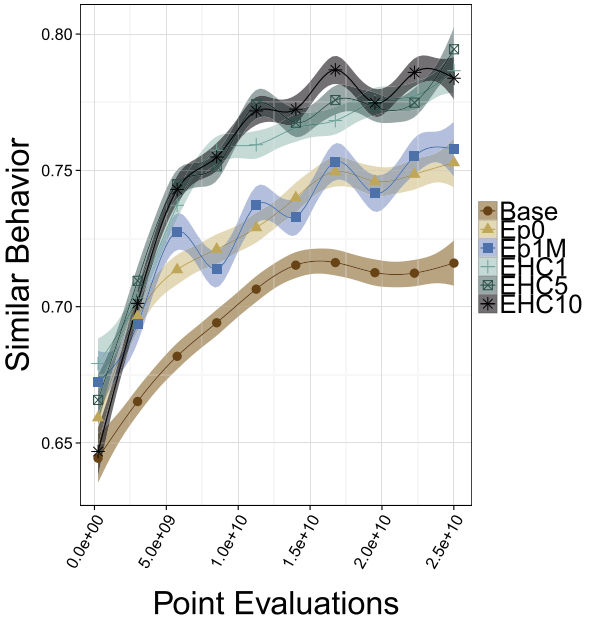

Fig. 12. *Similar Behavior* (fraction of unique output vectors in the population) for the Bacterial Respiration 2 problem.

homology (i.e. higher diversity) than *expressed* genotypes for every epigenetic treatment, thus demonstrating that genetic drift is indeed occurring in inactive sections of programs. For instance, we measured H for the Bacterial Respiration problem (state 2) for each treatment, the results of which are shown in Fig. 11. Homology in the expressed genome is more or less equivalent for all treatments. Despite having similar expressed genetic homologies, we find that behavioral similarity increases more quickly with the hill climbing methods (EHC1, EHC5, EHC10) than with Base, Ep0 or Ep1M, as shown in Fig. 12. To understand why the EHC methods converge the quickest semantically, recall that these methods only preserve epigenetic mutations that improve semantics, and are therefore more greedy than the other methods. Since the best methods for this problem converge on *Similar Behavior* the quickest, it appears that greedy topological search afforded by the EHC methods is an important factor in creating the improvements noted in our experiments. From the perspective of search, Figs. 11 and 12 imply that the EHC systems are exploiting neutral variation in the genome as well as improved reachability in the genotype-phenotype mapping provided by epigenetics, since H remains flat while *Similar Behavior* increases. Neutral variation is a property known to benefit other GP methods as well (Turner and Miller, 2015). In other words, the epigenetic systems converge on model behavior more quickly while preserving genetic diversity in the search space.

It is important to note that the smoothness of the fitness landscape of a problem will play a role in determining whether greedy methods like EHC are the best option. For example, we studied several program synthesis problems for which Ep1M provided better performance (La Cava et al., 2015). This could be due to rugged and/or deceptive fitness landscapes. Given that the EHC methods work best for the dynamic systems studied in this paper, it is likely that the fitness landscapes are less deceptive than those for program synthesis. It is also likely that similar systems in this domain have similar properties, and they should therefore benefit from the EHC variants of ELGP as well.

7. Conclusions

The results suggest that epigenetic local search is a significant addition to GP. We find that epigenetic methods, especially EHC methods, outperform a baseline implementation of GP in terms of fitness minimization, exact solutions, and equation intelligibility on textbook nonlinear ODE systems and randomly generated dynamic systems. Furthermore we show in comparison to other nonlinear approaches that ELGP is able to return concise and accurate models in three real-world applications. Our study of population diversity suggests that this performance improvement to GP is achieved by the epigenetic layer's ability to preserve diversity in the inactive sequences of genes while converging more quickly in semantic space. Although we have only considered epigenetic learning by mutation and hill climbing here, the results encourage further research into the use of epigenetic methods for structure optimization in GP, and motivate a focus on methods that improve the ability of GP to search equation topologies, in addition to constants.

Acknowledgments

We thank Nicholas McPhee and the Hampshire Computational Intelligence lab for helping improve this paper. This work is partially supported by the NSF-sponsored IGERT: Offshore Wind Energy Engineering, Environmental Science, and Policy (Grant no. 1068864), as well as Grant nos. 1017817, 1129139, and 1331283. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation Grant no. ACI-1053575 (Towns et al., 2014).

References

Arnaldo, I., Krawiec, K., O'Reilly, U.-M., 2014. Multiple regression genetic programming. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, ACM Press, pp. 879–886.

Banzhaf, W., 1994. Genotype-phenotype-mapping and neutral variation – a case study in genetic programming. In: Parallel Problem Solving from Nature (PPSN) III, Springer, pp. 322–332.

Billings, S.A., 2013. Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-temporal Domains. John Wiley & Sons.

Bongard, J., Lipson, H., 2007. Automated reverse engineering of nonlinear

dynamical systems. Proc. Natl. Acad. Sci. 104 (24), 9943–9948.
Brameier, M., Banzhaf, W., 2007. Linear Genetic Programming, vol. 1. Springer, 1 edition.
Cao, H., Kang, L., Chen, Y., Yu, J., 2000. Evolutionary modeling of systems of ordinary differential equations with genetic programming. Genet. Program. Evol. Mach. 1 (4), 309–337.
Chen, S., Billings, S.A., Luo, W., 1989. Orthogonal least squares methods and their application to non-linear system identification. Int. J. Control 50 (5), 1873–1896.
Cornforth, T.W., Lipson, H., 2013. Inference of hidden variables in systems of differential equations with genetic programming. Genet Program Evol. Mach. 14 (2), 155–190.
Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.-P. (Eds.), Parallel Problem Solving from Nature PPSN VI 1917. Springer, Berlin, Heidelberg, pp. 849–858.
Dias, B.G., Ressler, K.J., 2013. PACAP and the PAC1 receptor in post-traumatic stress disorder. Neuropsychopharmacology 38 (1), 245–246.
Dias, B.G., Ressler, K.J., 2014. Parental olfactory experience influences behavior and neural structure in subsequent generations. Nat. Neurosci. 17 (1), 89–96.
Ferreira, C., 2001. Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst. 13 (2), 87–129 arXiv:cs/0102027.
Fleming, P., Van Wingerden, J.-W., Wright, A.D., 2011. Comparing State-space Multivariable Controls to Multi-SISO Controls for Load Reduction of Drivetrain-coupled Modes on Wind Turbines through Field-testing: Preprint. National Renewable Energy Laboratory, National Wind Technology Center.
Fontana, A., 2011. Epigenetic tracking. In: Kampis, G., Karsai, I., Szathmáry, E. (Eds.), Advances in Artificial Life, Darwin Meets von Neumann, number 5777 in Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 10–17.
Giraud-Carrier, C., 2002. Unifying learning with evolution through Baldwinian evolution and Lamarckism. In: Advances in Computational Intelligence and Learning, Springer, pp. 159–168.
Gray, G.J., Murray-Smith, D.J., Li, Y., Sharman, K.C., Weinbrenner, T., 1998. Nonlinear model structure identification using genetic programming. Control Eng. Pract. 6 (11), 1341–1352.
Gregorčič, G., Lightbody, G., 2008. Nonlinear system identification: from multiple-model networks to Gaussian processes. Eng. Appl. Artif. Intell. 21 (7), 1035–1055.
Gruau, F., Whitley, D., 1993. Adding learning to the cellular development of neural networks: evolution and the Baldwin effect. Evolut. Comput. 1 (3), 213–233.
Haber, R., Unbehauen, H., 1990. Structure identification of nonlinear dynamic systems—a survey on input/output approaches. Automatica 26 (4), 651–677.
Holliday, R., 2006. Epigenetics: a historical overview. Epigenetics 1 (2), 76–80.
Iba, H., Sato, T., 1994. Genetic Programming with Local Hill-Climbing. Technical Report ETL-TR-94-4, Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba-city, Ibaraki, Japan, vol. 305.
Jablonka, E., Lamb, M.J., 2002. The changing concept of epigenetics. Ann. N.Y. Acad. Sci. 981 (1), 82–96.
Jeong, I.-K., Lee, J.-J., 1996. Adaptive simulated annealing genetic algorithm for system identification. Eng. Appl. Artif. Intell. 9 (5), 523–532.
Kaati, G., Bygren, L.O., Edvinsson, S., 2002. Cardiovascular and diabetes mortality determined by nutrition during parents' and grandparents' slow growth period. Eur. J. Human. Genet. 10 (11), 682.
Keijzer, M., 2003. Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (Eds.), Genetic Programming, number 2610 in Lecture Notes in Computer Science-Springer, Berlin, Heidelberg, pp. 70–82.
Keijzer, M., 2013. Push-forth: a light-weight, strongly-typed, stack-based genetic programming language. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '13 Companion, ACM, New York, NY, USA, pp. 1635–1640.
Kommenda, M., Kronberger, G., Winkler, S., Affenzeller, M., Wagner, S., 2013. Effects of constant optimization by nonlinear least squares minimization in symbolic regression. In: Blum, C., Alba, E., Bartz-Beielstein, T., Loiacono, D., Luna, F., Mehnen, J., Ochoa, G., Preuss, M., Tantar, E., Vanneschi, L. (Eds.), GECCO '13 Companion: Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion. ACM, Amsterdam, The Netherlands, pp. 1121–1128.
Kommenda, M., Kronberger, G., Affenzeller, M., Winkler, S.M., Burlacu, B., 2015. Evolving simple symbolic regression models by multi-objective genetic programming. In: Genetic Programming Theory and Practice, vol. XIV Genetic and Evolutionary Computation. Springer, Ann Arbor, MI.
Koza, J.R., 1992. Genetic Programming: on the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA.
Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A., Dunlap, F., 1997. Automated synthesis of analog electrical circuits by means of genetic programming. IEEE Trans. Evolut. Comput. 1 (2), 109–128.
La Cava, W., Danai, K., Spector, L., Fleming, P., Wright, A., Lackner, M., 2015. Automatic identification of wind turbine models using evolutionary multiobjective optimization. Renew. Energy.
La Cava, W.G., Danai, K., 2015. Gradient-based adaptation of continuous dynamic model structures. Int. J. Syst. Sci., 1–15.
La Cava, W., Helmuth, T., Spector, L., Danai, K., 2015. Genetic programming with epigenetic local search. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), ACM Press, pp. 1055–1062.
La Cava, W., Spector, L., Danai, K., Lackner, M., 2014. Evolving differential equations with developmental linear genetic programming and epigenetic hill climbing. In: Companion Proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO), ACM Press, pp. 141–142.
Langdon, W.B., 2012. Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!, Springer Science & Business Media, vol. 1.
Ljung, L., 1999. System Identification: Theory for the User, 2nd ed., Prentice Hall, vol. 1.
Ljung, L., 2007. System Identification Toolbox for Use with MATLAB®.
Madar, J., Abonyi, J., Szeifert, F., 2004. Genetic programming for system identification. In: Intelligent Systems Design and Applications (ISDA 2004) Conference, Budapest, Hungary.
Miller, J.F., Thomson, P., 2000. Cartesian genetic programming. In: Genetic Programming. Springer, pp. 121–132.
Narendra, K.S., Parthasarathy, K., 1990. Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Netw. 1 (1), 4–27.
Ni, X., Verhaegen, M., Krijgsman, A.J., Verbruggen, H.B., 1996. A new method for identification and control of nonlinear dynamic systems. Eng. Appl. Artif. Intell. 9 (3), 231–243.
Nordin, P., Francone, F., Banzhaf, W., 1995. Explicitly defined Introns and destructive crossover in genetic programming. In: Rosca, J.P. (Ed.), Proceedings of the Workshop on Genetic Programming: from Theory to Real-World Applications, Tahoe City, California, USA, pp. 6–22.
Perkis, T., 1994. Stack-based genetic programming. In: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence, Proceedings of the First IEEE Conference on. IEEE, pp. 148–153.
Rodriguez-Vazquez, K., Fonseca, C.M., Fleming, P.J., 2004. Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. IEEE Trans. Syst. Man, Cybern. - Part A: Syst. Hum. 34 (4), 531–545.
Rodríguez-Vázquez, K., Fleming, P.J., 2004. Evolution of mathematical models of chaotic systems based on multiobjective genetic programming. Knowl. Inf. Syst. 8 (2), 235–256.
Ross, B.J., 1999. A Lamarckian evolution strategy for genetic algorithms. Pr. Handb. Genet. Algorithms: Complex Coding Syst. 3, 1–16.
Sadollah, A., Eskandar, H., Yoo, D.G., Kim, J.H., 2015. Approximate solving of nonlinear ordinary differential equations using least square weight function and metaheuristic algorithms. Eng. Appl. Artif. Intell. 40, 117–132.
Schmidt, M., Lipson, H., 2009. Distilling free-form natural laws from experimental data. Science 324 (5923), 81–85.
Schmidt, M., Lipson, H., 2007. Comparison of tree and graph encodings as function of problem complexity. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, ACM, New York, NY, USA, pp. 1674–1679.
Schmidt, M., Lipson, H., 2011. Age-fitness pareto optimization. In: Genetic Programming Theory and Practice VIII, Springer, pp. 129–146.
Schmidt, M.D., 2011. Machine Science: Automated Modeling of Deterministic and Stochastic Dynamical Systems (Ph.D. thesis). Cornell University, Ithaca, NY, USA, AAI3484909.
Silverman, B.W., 1986. Density Estimation for Statistics and Data Analysis. CRC Press, vol. 26.
Smits, G.F., Kotanchek, M., 2005. Pareto-front exploitation in symbolic regression. In: Genetic Programming Theory and Practice II, Springer, pp. 283–299.
Spector, L., Robinson, A., 2002. Genetic programming and autoconstructive evolution with the push programming language. Genet. Program. Evol. Mach. 3 (1), 7–40.
Spector, L., Helmuth, T., 2013. Uniform linear transformation with repair and alternation in genetic programming. Genetic Programming Theory and Practice XI, page In preparation, Springer.
Strogatz, S.H., 2014. Nonlinear Dynamics and Chaos: with Applications to Physics, Biology, Chemistry, and Engineering. Westview Press.
Tanev, I., Yuta, K., 2008. Epigenetic programming: genetic programming incorporating epigenetic learning through modification of histones. Inf. Sci. 178 (23), 4469–4481.
Topchy, A., Punch, W.F., 2001. Faster genetic programming based on local gradient search of numeric leaf values. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 155–162.
Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G.D., Roskies, R., Scott, J.R., Wilkens-Diehr, N., 2014. XSEDE: accelerating scientific discovery. Comput. Sci. Eng. 16 (5), 62–74.
Turner, A.J., Miller, J.F., 2015. Neutral genetic drift: an investigation using Cartesian genetic programming. Genet Program Evol. Mach., 1–28.
Turner, B.M., 2000. Histone acetylation and an epigenetic code. Bioessays 22 (9), 836–845.
Vladislavleva, E., Smits, G., den Hertog, D., 2009. Order of nonlinearity as a complexity measure for models generated by symbolic regression via Pareto genetic programming. IEEE Trans. Evolut. Comput. 13 (2), 333–349.
Whitley, D., Gordon, V.S., Mathias, K., 1994. Lamarckian evolution, the Baldwin effect and function optimization. In: Parallel Problem Solving from Nature (PPSN) III, Springer, pp. 5–15.
Wigren, T., 2006. Recursive prediction error identification and scaling of non-linear state space models using a restricted black box parameterization. Automatica 42 (1), 159–168.
Wigren, T., 2010. Input-output data sets for development and benchmarking in nonlinear identification. Technical Reports from the Department of Information Technology, 20:2010–020, 2010. Data sets available: ⟨http://www.it.uu.se/research/publications/reports/2010-020/NonlinearData.zip⟩.
Wigren, T., Schoukens, J., 2013. Three free data sets for development and benchmarking in nonlinear system identification. In: Proc. 2013 Eur. Control Conf. (ECC2013), pp. 17–19.
Wu, A.S., Lindsay, R.K., 1995. Empirical studies of the genetic algorithm with non-coding segments. Evolut. Comput. 3 (2), 121–147.
Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: improving the strength Pareto evolutionary algorithm. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK).