Chapter 3
Inheritable Epigenetics in Genetic Programming

William La Cava and Lee Spector

3.1 Introduction

In 1815, Lamarck postulated that organisms acquired adaptations from their environments during their lifetime and these adaptations were passed along to the offspring they produced. Four decades later, Darwin showed evidence that organisms instead evolve traits over millions of years through a combination of random mutation, natural selection, and the transmission of genetic information to their progeny, rather than through lifetime adaptations (Darwin 1872). Four decades after that, Baldwin reconciled the two ideas by asserting that despite the inheritance of purely genetic material, adaptive changes during an organism's life affected selection pressures and thus reproductive dynamics, ensuring that the emergent adaptive ability of a genotype could be selected for without the information explicitly being transferred in addition to DNA.

The subsequent discovery of somatic and germ cell architecture and the biological mechanisms for genetic inheritance caused Lamarck's ideas to be discredited in biology. This did not stop researchers from studying the incorporation of Lamarckism and Baldwinian evolution into genetic algorithms (GAs) (Gruau and Whitley 1993; Whitley et al. 1994; Ross 1999; Giraud-Carrier 2002). In Lamarckian evolution for GAs, a local search mechanism is implemented to update the population genomes each generation. The Baldwin effect is achieved using the same local search to update the population fitness landscape, but the actual genotype changes are discarded. Whitley showed that both methods were quite useful for improving results for function optimization problems (Whitley et al. 1994). He noted that Lamarckism tended

W. La Cava (✉)
Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst, MA USA
e-mail: wlacava@umass.edu

L. Spector
School of Cognitive Science, Hampshire College, Amherst, MA 01002-3359, USA
e-mail: lspector@hampshire.edu

R. Riolo et al. (eds.), *Genetic Programming Theory and Practice XII*,
Genetic and Evolutionary Computation, DOI 10.1007/978-3-319-16030-6_3

to improve the speed of convergence but for some cases became stuck in the same local optima as the standard GA, whereas the Baldwinian evolution strategy was better able to find global optima.

Various Lamarckian updating methods have been implemented for genetic programming (GP) as well, such as equation tree snipping (Bongard and Lipson 2007), reinforcement learning (Mingo and Aler 2007), and parameter updating (Iba 2008; Topchy and Punch 2001). In symbolic regression in particular, concurrent parameter updating, either by stochastic hill climbing or gradient methods, is common due to the vast floating-point search space, and it has been shown to improve convergence on a number of benchmark symbolic regression problems (Kommenda et al. 2013).

Researchers have been able to exploit the fact that, unlike in nature, little or no phenotypic-genotypic mapping has to occur in the computational scheme since the system being evolved is either identical to the genotype or the mapping of phenotypic traits to genotype is straightforward. Previously it was assumed that the Lamarckian implementation was extra-biological because phenotypic adaptations in nature did not have a known physical mechanism for influencing their genotypic origins (Ross 1999). Today, however, physical mechanisms are known to exist and have been demonstrated in many studies. The studies constitute the growing field of epigenetics, a term that refers broadly to the ways in which gene expressions are regulated and inherited (Jablonka and Lamb 2002; Holliday 2006). Recent studies have not only shown that environmental factors influence gene expression in organisms (Dias and Ressler 2013), but also that epigenetic mechanisms may be inheritable (Turner 2000; Kaati et al. 2002; Pogribny et al. 2004; Dias and Ressler. 2014).

We present a GP method that captures this understanding of epigenetics as a layer of environmentally influenced, evolving gene regulation that interacts with the genotype to produce the phenotype. This system captures the advantages of Lamarckian updating without changing the genotype, and yet preserves inheritable phenotypic improvements in offspring, unlike Baldwinian evolution.

3.2 Background

3.2.1 GP Representation

Biological systems benefit from transmitting and evolving structurally complex systems at the more flexible genome level. The key to preserving this flexibility in GP is to create a mapping from the genotype, i.e. the computer encoding, to the phenotype, e.g. the candidate equation[1]. In this way the genotypic search can be free to vary through evolutionary processes and still produce constrained phenotypes. As shown in Ryan (1996), lower constraints on evolutionary search generally improve results. This is the goal of developmental GP approaches such as genotype-phenotype mapping and gene expression programming (Banzhaf 1994; Ferreira 2001).

[1] Note that these definitions distinguish between the program, the resulting equation, and its fitness, unlike in traditional GP.

Classical GP (Koza 1992) represents individuals using tree structures, and most practitioners of symbolic regression follow this trend today, as White discovered in his community survey (White et al. 2012). In addition to tree representations, other methods have been proposed, for example stack-based GP (Salman et al. 1985; Spector 2001; Ferreira 2001), linear GP and directed acyclic graphs (Brameier and Banzhaf 2007; Schmidt and Lipson 2007), tree adjunct grammars (Hoai et al. 2002), and Cartesian GP (Miller and Thomson 2000). Despite the succinct representation of nested equations that tree structures provide, they have disadvantages when applied to evolutionary computation. For example, the tree structure makes it difficult to deliver uniform variation among instructions because of the effect that the size and shape of the trees have on the probability of change wrought by standard mutation and crossover. Whereas in nature chromosomal crossover occurs on homologous or nearly homologous sections of chromosomes between parents in a somewhat (but not purely) uniform manner, standard GP crossover consists of the swapping of random subtrees between parents. In standard GP, subtree mutation also occurs at a random node location. Therefore the probability of mutation and crossover for terminals and nodes in a tree is a positive function of its depth. With linear GP, this probability can be made more uniform. Motivated by evidence that uniformity improves evolutionary search (Page et al. 1999), the ULTRA operator (Spector and Helmuth 2013) was developed that converts nested expressions into linear representations, applies quasi-uniform crossover, repairs the program parentheses, and converts the representation back into a tree structure. The need for the 'R' (signifying repair) in the ULTRA operator highlights a second disadvantage of tree representations: random manipulation of tree structures at the level of instructions and literals can easily make them syntactically invalid, and therefore controls must be in place to ensure that operations such as mutation, crossover and initialization result in executable programs.

In our work we present a developmental linear genetic programming tool that evolves equations with an instruction set that is syntax-free with respect to program validity by using the principles of stack-based, post-fix equation encoding with all instruction ordering constraints removed. In this way, genotypes are unconstrained except by initial size during the run, and this opens the door for our investigation of epigenetics. In most GP systems, it is possible for distinct genotypes to result in the same phenotype, resulting in higher search flexibility. By applying epigenetic logic to the genotype, we are able to achieve the reverse as well: different phenotypic expression from identical genotypes. This could provide a path for increased diversity in a population and avoid premature convergence related to bottle-necking, i.e. genotypic convergence. We will investigate how this affects the performance of genetic programming in symbolic regression. The proposed epigenetic methods can also be thought of as a new local search extension to GP that provides a mechanism for modifying equation form by affecting phenotypic development.

3.2.2 Epigenetics

There are two main mechanisms by which epigenetics take place in mammalian DNA: cytosine methylation (Jones and Takai 2001) and histone modification (Turner 2000). Methylation provides a mechanism for silencing portions of genetic code, and as such is able to determine whether certain genes are expressed during transcription. Histones, meanwhile, are structures around which base pairs of DNA are wrapped, and their modification affects the winding of these base pairs, which provide an epigenetic mechanism by affecting the accessibility of genes for transcription.

These processes result in clusters of genetic material that are not expressed in the phenotype (a.k.a. non-coding segments or introns). Introns are studied in the genetic programming world, and initially were linked to bloat, a phenomenon in which programs grow very large, become resistant to behavioral change, and drain computer resources without improvement. However, there have been several studies on the effects of non-coding segments in evolutionary algorithms (EAs) that find that moderate levels of introns can improve EA performance by reducing the destructive effect of crossover operations while maintaining blocks of effective code (Nordin et al. 1995; Brameier and Banzhaf 2007). In Nordin et al. (1995) and Brameier and Banzhaf (2007), introns are either explicitly declared in place of genes or defined as such afterwards by observing their behavior. In our work, we instead impose an epigenetic condition on each instruction that functions as an on/off switch to determine whether or not the instruction will execute within the genotype. Therefore, silenced genes do not affect computation time during genotype to phenotype conversion, but are present as structural elements during mutation and crossover. Epigenetic activation and silencing is learned each generation using a stochastic hill climber and co-evolves with the corresponding genotypes.

There has been work to simulate epigenetics in GP: Tanev (Tanev and Yuta 2008) developed a genetic programming approach that simulated histone modification for use in a predator-prey problem. This approach did not include inheritance of epigenetic properties during evolution, which Tanev considered to be unrealistic at the histone level. Studies of artificial ontogeny (Bongard and Pfeifer 2001; Fontana 2011) encompass some of the aspects of epigenetics, particularly the evolution of phenotype control from a decoupled genotype. Still, an epigenetic approach has not been presented that attempts to model two salient characteristics: its ability to update based on environmental changes, and its ability to be inherited.

We make the assumption that epigenetic traits are inheritable, and focus on doing so in a generic way that can be readily applied to any genetic programming system. While the method is generic, a flat, syntax-free representation like the one presented in this paper is advantageous for epigenetic switching because genotypic regulation can be applied uniformly and easily since it does not require syntactic repair. With other representations, the probability of epigenetic modifications may not be uniform, and they may have to undergo a repair step to guarantee execution.

Many researchers in the field of GP see the incorporation of meta-genetic biological functions as an open issue in GP, with O'Neill et al. noting that "... it can be safely predicted that epigenetic effects will be important if GP will adopt development as a scalability mechanism." (ONeill et al. 2010)

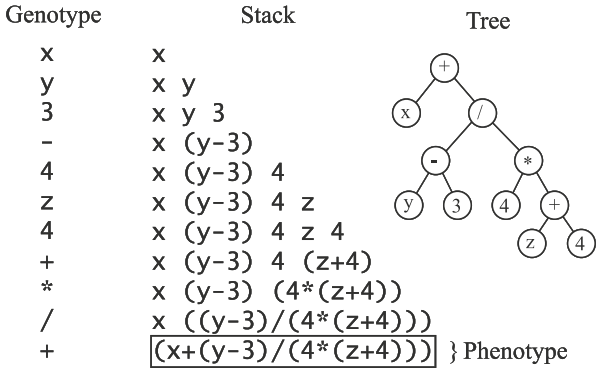

Fig. 3.1 Example encoding of $x + \frac{y-3}{4(z+4)}$ with the steps of stack execution and equivalent tree representation

3.3 Methods

3.3.1 Developmental Linear Genetic Programming

The GP system created for this research is called Develep and the source code is available online (La Cava 2014a). We represent programs as linear genotypes written in Reverse Polish Notation (RPN) or "post-fix" notation, as demonstrated in Figure 3.1. The implementation functions by pushing and pulling floating point numbers on and off of the stack. For instance, a number or variable instruction will push a floating point value to the stack, while a binary (i.e. arity 2) operation will pull two numbers off of the stack, perform its operation on them, and push a new value back to the stack. This basic representation scheme is used in other GP systems (Salman et al. 1985; Ferreira 2001; Spector and Robinson 2002).

In order to guarantee execution safety in the context of arbitrary execution order, the instructions are specialized to handle situations in which the stack does not have the correct number of elements for an operation to occur. For example, binary functions are ignored if the stack is not at least two elements long. At the end of genotype execution, the top element of the stack constitutes the equation, the phenotype, that is then used for fitness assessment. Therefore execution is safe with respect to the resultant stack length. We distinguish between the genotype, that builds the stack, and the phenotype, that is the top element of the resultant stack.

3.3.2 Epigenesis

3.3.2.1 Epigenetic Hill Climber

Epigenesis is achieved by creating binary switches associated with the instructions comprising the genotypes of each individual, and the array of these switches is referred to as an epiline. During genotype execution, only instructions from the genotype with a true value in the corresponding epiline are executed. Individuals can

be initialized with any combination of active and inactive epiline values, and each generation the population undergoes one iteration of epigenetic hill climbing (EHC) to optimize the epiline.

```
gen ⟵ genotype of individual
epi ⟵ epigenetic line of individual
phen ⟵ phenotype equation of individual
L ⟵ length of genotype
sr ⟵ switching rate
for number of hill climbing iterations do
    epiTemp ⟵ epi
    for i ∈ L do
        if rand() < sr then
            /* flip the on/off state of marked indices in the
               epiline                                          */
            epiTemp(i) ⟵ !epi(i)
        end
    end
    /* get equation from genotype with updated gene expression
       */
    phenTemp ⟵ GenToPhen(gen,epiTemp)
                                          /* update equation   */
    if fitness(phenTemp) < fitness(phen) then
        epi ⟵ epiTemp
        phen ⟵ phenTemp
                                          /* secondary size metric */
    else if fitness(phenTemp) == fitness(phen) and size(phenTemp) < size(phen) then
        epi ⟵ epiTemp
        phen ⟵ phenTemp
    else
    end
end
```

Algorithm 1 shows the EHC algorithm for updating the epigenetic properties of an individual in the population. During the hill climbing process, a small percentage of epiline values are switched on or off and the genotype is re-executed with the new epiline. For example, compare the phenotype of Fig. 3.1 to that generated with epigenetic switching in Fig. 3.2. A different equation can be expressed by an identical genotype in this way. The resulting equation is evaluated and kept (along with the new epiline) if it results in a better fitness for the individual, or if it results in a smaller equation size without changing the fitness. Equation size is measured by the number of characters in the equation string. Using string length is a simple but crude measure of equation size, and while more sophisticated measures could be used, calculating string length is very lightweight and inherently penalizes inefficient equation representations, such as having lots of nesting or many separate constants in the phenotype.

The EHC is applied after fitness evaluation and before selection. Figure 3.3 shows the implementation within the Develep instruction set. After an initial fitness

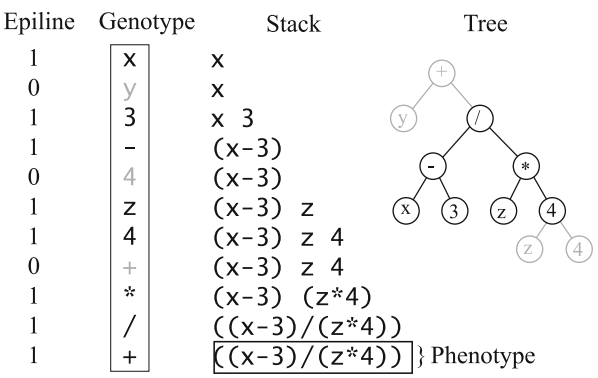

Fig. 3.2 Epigenetics added to the original encoding of $f = x + \frac{y-3}{4(z+4)}$, which in this example results in $f = \frac{x-3}{(z*4)}$ Note that this requires the replacement of the '+' operator in the tree representation with the 4 operand. Thus a repair step would be required for tree implementation

evaluation, one iteration of epigenetic hill climbing is applied. The better fitness value and corresponding epiline and phenotype are kept. The EHC is therefore a local optimization scheme within the scope of genetic material already available to the individual. It can both decrease and increase the expressed length of the genotype. Silenced genes reduce the computational effort of developing the phenotype and evaluating the fitness of the genotype by lowering the number of point evaluations.

3.3.2.2 Evolution of Epiline

The epiline values for an individual are connected to specific genes. During crossover, the child inherits its parents' genes along with the genes' epigenetic states. In the case of swap mutation, the new gene's epigenetic state is chosen probabilistically from the chosen percent of active genes in the initial population (i.e. if the population is

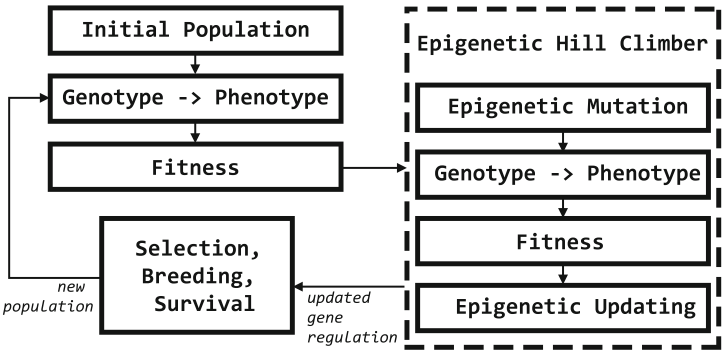

Fig. 3.3 Block diagram of developmental epigenetic programming. After fitness evaluation and before selection, the population undergoes an iteration of epigenetic hill climbing, represented by the *dotted lines* on the *right*. The population then undergoes the typical process of selection, breeding and survival to produce the next generation, and the process is repeated, as shown on the *left*

initialized to be 50 % active, a new gene mutation will have a 50 % chance of being active as well).

It should be noted that modeling the epigenetic inheritance after genetic inheritance is biologically questionable. While it is generally agreed that there are sets of imprinted genes whose epigenetic states are inherited in offspring, there is a reprogramming stage in embryonic development during which many epigenetic values are erased and reapplied. The way that this reprogramming functions and whether it is itself inherited is not well understood. We assume genotypic analog for epigenetic inheritance in this work due to the ease with which we can embed the epigenesis in our representation, and the fact that these values go through a small amount of reprogramming thanks to the EHC.

3.3.2.3 Evolutionary Parameters

Crossover and mutation are used as genetic operators, with a 90 % rate of crossover and 10 % rate of mutation. During crossover, a random point is picked in each parent, and the tails of the two parents are swapped (a.k.a. one-point crossover). The mutation operator is a pointwise operator inspired by Bongard's hill climber (Bongard and Lipson 2007). It can change or delete a randomly selected gene. Whereas all other instructions are replaced randomly from the instruction set, operand instructions (e.g. x, y, 1.73, etc.) are mutated by changing the argument they contain. If the argument is a variable, it is swapped for a randomly picked one. If the argument is a constant, the constant is adjusted by 0 mean Gaussian noise with standard deviation equal to half the constant magnitude or replaced with a random constant, with equal probability. Genes are selected for mutation with a 10 % probability. This probability may seem high for a standard mutation operator, but due to the typing constraints described above, is believed to be appropriate.

We use one of two modern evolutionary methods in our trials: deterministic crowding (DC) or age-fitness Pareto survival (AFP). In deterministic crowding (Mahfoud 1995), children replace the parent with the smallest phenotypic Levenshtein distance if and only if they have a lower fitness than the parent. In age-fitness Pareto survival (Schmidt and Lipson 2011), each individual has an age equal to the number of generations its oldest genes have been in the population. Each generation, a whole separate population of offspring and one new individual are created. These individuals compete with the current population in a tournament of size two that compares age and fitness dominance, and the winners survive to the next generation. In both DC and AFP, parents are randomly selected to produce children.

3.4 Applications

We apply the genetic programming system to a two-state differential equation problem and two benchmark symbolic regression problems. The first two problems consist of the partitioned (Bongard and Lipson 2007) states of the Lotka-Volterra interspecies competition model, defined as

Table 3.1 Runtime settings

Setting	Lotka-volterra	Pagie-1	Nguyen-7
Terminal set	{+, −,*,/,R[−1.0,1.0],x,y}	{+, −,*,/,1.0,x}	{+, −,*,/,exp,log,1.0,x}
Initial program length	[3, 50]	[10, 100]	[10, 100]
Method	DC[a]	AFP[b]	AFP[b]
Pop size	1000	1000	1000
Max generations	5000	5000	5000
Initial % active genes	100	50	50

Brackets indicate ranges of values picked uniformly
R ephemeral random constants
[a]Deterministic Crowding
[b]Age-Fitness Pareto Survival

$$\dot{x} = 3x - 2xy - x^2 \tag{3.1}$$

$$\dot{y} = 2y - xy - y^2 \tag{3.2}$$

The benchmark symbolic regression were chosen from the set of recommended problems emerging from a GP community survey (White et al. 2012). We used the Pagie-1 and Nguyen-7 problems. The Pagie-1 problem is a two-variable system defined as

$$f(x, y) = \frac{1}{1 + x^{-4}} + \frac{1}{1 + y^{-4}} \tag{3.3}$$

The Nguyen-7 problem is defined by the single-variable equation

$$f(x) = log(x + 1) + log(x^2 + 1) \tag{3.4}$$

The settings for all the trials are shown in Table 3.1. The differential equation problems were run using deterministic crowding, whereas the benchmark problems were solved using age-fitness Pareto survival.

Ephemeral random constants were included in the terminal set for the Lotka-Volterra problems, and the Nguyen-7 addded the exp and log functions. For Lotka-Volterra, all genes were active in the initial population. After running a parameter variation study that showed higher rates of beneficial crossover with a mix of active and inactive genes in the inital population (La Cava et al. 2014b), a starting value of 50% active genes was used for the Pagie-1 and Nguyen-7 problems.

We measured a successful run as one achieving

$$\sum_{i=1}^{n} |y^*(i) - \hat{y}(i)| < 0.0001$$

where y^* is the target output, $\hat{y}$ is the equation output, and n is the number of data points. The fitness metric F used during the runs includes the coefficient of determination R^2 and is defined as

$$F = \frac{\frac{1}{n}\sum_{i=1}^{n} |y^*(i) - \hat{y}(i)|}{R^2} \tag{3.5}$$

Table 3.2 Performance comparisons

Problem	Trials	Method	Success rate (%)	MBF	Point evaluations per fitness case	Mean effective size
Lotka-Volterra $\dot{x}$	50	DLGP	100	0.000	*2.7430E07*	*29.63*
	50	DLGP+EHC	100	0.000	*2.0655E07*	*24.69*
Lotka-Volterra $\dot{y}$	50	DLGP	100	0.000	2.5763E07	*30.3595*
	50	DLGP+EHC	100	0.000	2.2529E07	*25.1297*
Pagie-1	100	DLGP	*13*	*0.086*	*3.8605E08*	*68.7337*
	100	DLGP + EHC	*27*	*0.054*	*3.8678E08*	*40.3222*
Nguyen-7	50	DLGP	*72*	*0.0021*	*2.2360E08*	*68.97*
	50	DLGP+EHC	*100*	*0.000*	*2.2781E07*	*20.2493*

Results in *italic* are significant to $p < 0.05$, where p is the non-parametric ranked t-test. (Wineberg and Christensen 1994)

where

$$R^2 = \frac{cov(y^*, \hat{y})^2}{var(y^*)var(\hat{y})} \tag{3.6}$$

For these examples, every successful run was an exact match to the target equation form, with parameters exact to floating-point zero.

A parameter hill climber was used once per generation on each individual in order to perform local search of constant values in the Lotka-Volterra problems. The parameter hill climber perturbs each constant value in the expressed genotype with 0 mean Gaussian noise with standard deviation equal to 10 % of the magnitude of the constant, and keeps the changes if they improve fitness. Likewise, the EHC was run for one iteration each generation with a switching rate of 10 %.

3.5 Results and Discussion

The results are summarized in Table 3.2. We compare results by success rate (exact solution), mean best fitness (MBF), mean number of total point evaluations in the trials, and mean effective program length (number of active instructions) during the trial. The EHC increased success rate on the Pagie-1 problem from 13 to 27 %, and increased success rate on the Nguyen-7 problem from 72 to 100 %; meanwhile, standard DLGP and DLGP+EHC both solved the Lotka-Volterra problem 100 % of the time.

Despite increasing the number of fitness evaluations per generation by implementing the EHC, we found that the total number of point evaluations during a run actually decreased for most problems, with statistically significant decreases in the Lotka-Volterra and Nguyen-7 problem. This is due to a combination of improved success rate (for Nguyen-7) and the lower effective program length. For the Pagie-1

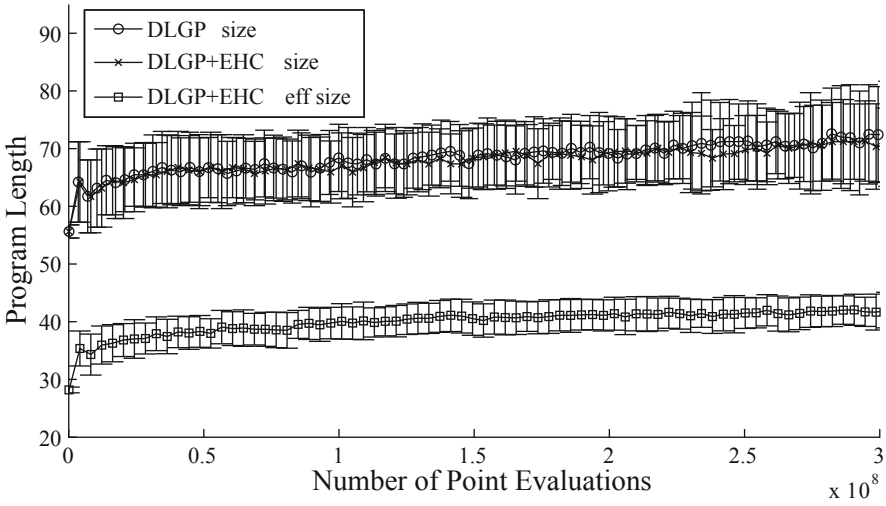

Fig. 3.4 Program size comparison during evolution of Pagie-1, averaged over the trials of each method. Both total and effective length are shown for the regular *DLGP* method and the additional *EHC* implementation

problem there was approximately a 2 % increase in total point evaluations, likely due to the fact that most trials went to the maximum generations without finding a solution. While it is difficult to make a direct comparison to previously published results due to the variability in experimental setup and termination criteria, it is noted that most algorithms do not find (or do not report) as many exact solutions to the Pagie-1 problem (Pagie and Hogeweg 1997; Kommenda et al. 2013; Spector and Helmuth 2013) [eg] or the Nguyen-7 problem (Uy et al. 2011; Krawiec and Pawlak, 2013), especially when using "standard" GP.

We also found that the EHC provided good size control during evolution, as shown in Figs 3.4 and 3.5. While the total genotypic size remains similar to DLGP, with EHC activated the effective size of the programs was 17 % shorter for each Lotka-Volterra state, 41 % shorter for Pagie-1, and 70 % shorter for Nguyen-7. For the Lotka Volterra cases, the smaller initial program sizes and 0 % inactive genes may explain why the size difference is less pronounced. Furthermore, as Fig. 3.5 shows, the EHC addition resulted in final solutions with less bloat, meaning that the expressed genotype of the solution was closer to the smallest possible size by which the exact solutions could be represented.

3.6 Conclusion

We have demonstrated a straightforward way to incorporate some of the key features of epigenetics into linear genetic programming in order to improve symbolic regression performance. We represented two characteristics of epigenesis in this

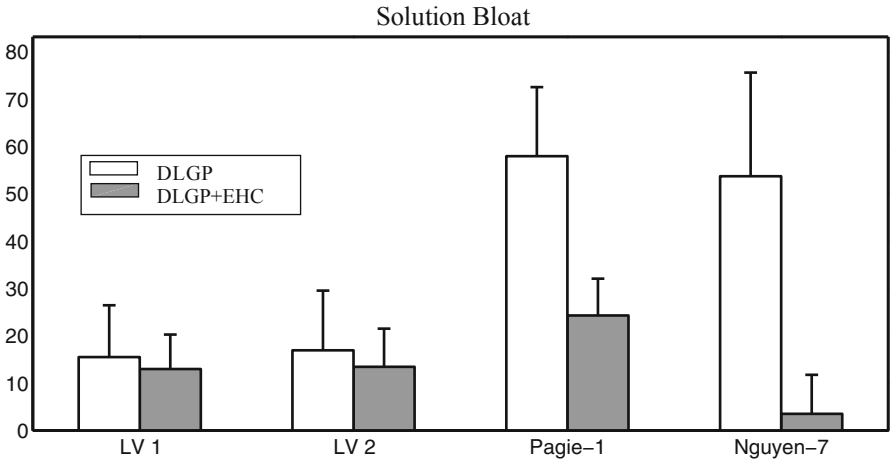

Fig. 3.5 Comparison of solution bloat, which is the difference in size of the solution program and the smallest possible solution program for the exact solution

implementation: (1) dependence on environmental factors by use of the EHC, and (2) inheritability by evolution of epilines with their corresponding genotypes. Unlike previous methods, our system allows offspring to inherit both the learned phenotypic traits of their parents as well as the genotypic underpinning. With this system we demonstrate higher success rates and lower solution bloat for a number of symbolic regression problems, with equivalent or lower computational effort required. This suggests that the epigenetic implementation is able to capitalize on the benefits of Lamarckism (fast convergence) and Baldwinian evolution (finding global optima). We hope this work will provide the basis for further investigation into how epigenetic learning and evolution can interact to improve genetic programming for many applications. Namely, further work should address various levels of epigenetic inheritability, as well as the contributions of environmental factors or inheritance to the improvement in success.

Acknowledgements The authors would like to thank Thomas Helmuth for his insightful feedback and Professor Kourosh Danai for his support of this research, as well as the members of the Hampshire Computational Intelligence Laboratory. This work is partially supported by the NSF-sponsored IGERT: Offshore Wind Energy Engineering, Environmental Science, and Policy (Grant Number 1068864), as well as Grant No. 1017817. 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.

References

Banzhaf W (1994) Genotype-phenotype-mapping and neutral variationa case study in genetic programming. Parallel problem solving from naturePPSN III. Springer, p 322–332. http://link.springer.com/chapter/10.1007/3-540-58484-6-276

Bongard J, Lipson H (2007) Automated reverse engineering of nonlinear dynamical systems. Proc Natl Acad Sci USA 104(24):9943–9948. http://www.pnas.org/content/104/24/9943.short

Bongard JC, Pfeifer R (2001) Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. Proceedings of the genetic and evolutionary computation conference, p 829–836. http://www.cems.uvm.edu/jbongard/papers/geccoBongard2001.pdf

Brameier M, Banzhaf W (2007) Linear genetic programming, vol 1, 1st edn. Springer Berlin

Darwin C (1872) The origin of species by means of natural selection: or, the preservation of favoured races in the struggle for life and the descent of man and selection in relation to sex. Modern Library, London

Dias BG, Ressler KJ (2013) PACAP and the PAC1 receptor in post-traumatic stress disorder. Neuropsychopharmacology 38(1):245–246. doi:10.1038/npp.2012.147. http://www.nature.com/npp/journal/v38/n1/full/npp2012147a.html

Dias BG, Ressler KJ (2014) Parental olfactory experience influences behavior and neural structure in subsequent generations. Nat Neurosci 17(1):89–96. doi:10.1038/nn. http://www.nature.com/neuro/journal/v17/n1/full/nn.3594.html

Ferreira C (2001) Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst 13(2):87–129. arXiv:cs/0102027. http://arxiv.org/abs/cs/0102027

Fontana A (2011) Epigenetic tracking: biological implications. In: Kampis G, Karsai I, Szathmry E (eds) Advances in artificial life. Darwin Meets von Neumann, no. 5777 in Lecture notes in computer science. Springer, Berlin, pp 10–17. http://link.springer.com/chapter/10.1007/978-3-642-21283-3-2

Giraud-Carrier C (2002) Unifying learning with evolution through baldwinian evolution and lamarckism. In Advances in Computational Intelligence and Learning (pp. 159–168). Springer Netherlands

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. doi:10.1162/evco.1993.1.3.213. http://dx.doi.org/http://dx.doi.org/10.1162/evco.1993.1.3.213

Hoai NX, McKay RI, Essam D, Chau R (2002) Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: the comparative results. Evolutionary computation, 2002. CEC'02. Proceedings of the 2002 Congress on, IEEE, vol 2, p 1326–1331. http://ieeexplore.ieee.org/xpls/abs-all.jsp?arnumber=1004435

Holliday R (2006) Epigenetics: a historical overview. Epigenetics 1(2). http://www.landesbioscience.com/journals/epigenetics/holliday EPI1-2.pdf

Iba H (2008) Inference of differential equation models by genetic programming. Inf Sci 178(23):4453–4468. doi:10.1016/j.ins.2008.07.029. (special Section: Genetic and Evolutionary Computing)

Jablonka E, Lamb MJ (2002) The changing concept of epigenetics. Ann NY Acad Sci 981(1):82–96. http://onlinelibrary.wiley.com/doi/10.1111/j.1749-6632.2002.t b04913.x/full

Jones PA, Takai D (2001) The role of DNA methylation in mammalian epigenetics. Science 293(5532):1068–1070. doi:10.1126/science.1063852. http://www.sciencemag.org/content/293/5532/1068, PMID: 11498573

Kaati G, Bygren LO, Edvinsson S (2002) Cardiovascular and diabetes mortality determined by nutrition during parents' and grandparents' slow growth period. Eur J Hum Genet 10(11):682

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, p 1121–1128. doi:10.1145/2464576.2482691

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge

Krawiec K, Pawlak T (2013) Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. Genet Program Evolvable Mach 14(1):31–63. doi:10.1007/s10710-012-9172-7 . http://link.springer.com/article/10.1007/s10710-012-9172–7

La Cava W (2014a) Develep (Version 1.0.0) [software]. doi:10.5281/zenodo.9824. Retrieved from https://zenodo.org/record/9824

La Cava W, Spector L, Danai K, Lackner M (2014b) Evolving differential equations with developmental linear genetic programming and epigenetic hill climbing. GECCO '14: companion publication of the 2014 genetic and evolutionary computation conference. ACM. doi: http://dx.doi.org/10.1145/2598394.2598491

Mahfoud SW (1995) Niching methods for genetic algorithms. PhD thesis, University of Illinois at Urbana-Champaign

Miller JF, Thomson P (2000) Cartesian genetic programming. Genetic programming. Springer Berlin Heidelberg, p 121–132. http://link.springer.com/chapter/10.1007/978-3-540-46239-2-9

Mingo J, Aler R (2007) Grammatical evolution guided by reinforcement. IEEE congress on evolutionary computation . CEC 2007, pp 1475–1482. 10.1109/CEC.2007.4424646

Nordin P, Francone F, Banzhaf W (1995) Explicitly defined introns and destructive crossover in genetic programming. In: Rosca JP (ed) Proceedings of the workshop on genetic programming: from theory to real-world applications. Tahoe City, p 6–22. http://web.cs.mun.ca/banzhaf/papers/ML95.pdf

ONeill M, Vanneschi L, Gustafson S, Banzhaf W (2010) Open issues in genetic programming. Genet Program Evolvable Mach 11(3–4):339–363. doi:10.1007/s10710-010-9113-2. http://link.springer.com/article/10.1007/s10710-010-9113–2

Page J, Poli R, Langdon WB (1999) Smooth uniform crossover with smooth point mutation in genetic programming: a preliminary study. Genetic programming, proceedings of EuroGP'99, vol 1598 of LNCS, Springer-Verlag, pp 39–49

Pagie L, Hogeweg P (1997) Evolutionary consequences of coevolving targets. Evol Comput 5(4):401–418. http://www.mitpressjournals.org/doi/abs/10.1162/evco.1997.5.4.401

Pogribny I, Raiche J, Slovack M, Kovalchuk O (2004) Dose-dependence, sex- and tissue-specificity, and persistence of radiation-induced genomic DNA methylation changes. Biochem Biophys Res Commun 320(4):1253–1261. doi: 10.1016/j.bbrc.2004.06.081. http://www.sciencedirect.com/science/article/pii/S0006291X 040 13208

Ross BJ (1999) A lamarckian evolution strategy for genetic algorithms. Pract Handb Genet Algorithms Complex Codin Syst 3:1–16

Ryan C (1996) Reducing premature convergence in evolutionary algorithms. PhD thesis, National University of Ireland

Salman WP, Tisserand O, Toulout B, Stewart M (1985) Forth. Springer-Verlag, New York

Schmidt M, Lipson H (2007) Comparison of tree and graph encodings as function of problem complexity. Proceedings of the 9th annual conference on genetic and evolutionary computation, ACM, New York, GECCO '07, p 674–1679. doi:10.1145/1276958.1277288. http://doi.acm.org/10.1145/1276958.1277288

Schmidt M, Lipson H (2011) Age-fitness pareto optimization. Genetic programming theory and practice VIII, p 129–146. Springer New York. http://link.springer.com/chapter/10.1007/978-1-4419-7747-2–8

Spector L (2001) Autoconstructive evolution: Push, pushGP, and pushpop. Proceedings of the genetic and evolutionary computation conference GECCO-2001, p 137–146

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. http://people.cs.umass.edu/thelmuth/Pubs/ultra-gptp-2013-preprint.pdf

Spector L, Robinson A (2002) Genetic programming and autoconstructive evolution with the push programming language. Genet Program Evolv Mach 3(1):7–40. http://link.springer.com/article/10.1023/A:1014538503543

Tanev I, Yuta K (2008) Epigenetic programming:genetic programming incorporating epigenetic learning through modification of histones. Inf Sci 178(23):4469–4481.doi: 10.1016/j.ins.2008.07.027. http://linkinghub.elsevier.com/retrieve/pii/S0020025508002880

Topchy A, Punch WF (2001) Faster genetic programming based on local gradient search of numeric leaf values. Proceedings of the genetic and evolutionary computation conference GECCO-2001, p 155–162. http://garage.cse.msu.edu/papers/GARAGe01-07-01.pdf

Turner BM (2000) Histone acetylation and an epigenetic code. Bioessays 22(9):836–845

Uy NQ, Hoai NX, ONeill M, McKay RI, Galvn-Lpez E (2011) Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genet Program Evolv Mach 12(2):91–119. http://link.springer.com/article/10.1007/s10710-010-9121-2

White DR, McDermott J, Castelli M, Manzoni L, Goldman BW, Kronberger G, Jakowski W, OReilly UM, Luke S (2012) Better GP benchmarks: community survey results and proposals. Genet Program Evolv Mach 14(1):3–29. doi:10.1007/s10710-012-9177-2. http://link.springer.com/10.1007/s10710-012-9177–2

Whitley D, Gordon VS, Mathias K (1994) Lamarckian evolution, the baldwin effect and function optimization. Parallel problem solving from NaturePPSN III. Springer Berlin Heidelberg, p 5–15. http://link.springer.com/chapter/10.1007/3-540-58484-6-245

Wineberg M, Christensen S (2004) An introduction to statistics for EC experimental analysis. http://www.cis.uoguelph.ca/wineberg/publications/ECStat2004.pdf. Accessed 10 June 2014

William La Cava is a PhD student at University of Massachusetts, Amherst. He received his B.S. and M.Eng. from Cornell University in 2009 and 2010, and went on to work at the National Wind Technology Center in Boulder, Colorado. William is interested in automatic system identification and automatic control design for complex systems. As an NSF student fellow in the UMass IGERT Offshore Wind Energy Program, he is developing symbolic regression methods to create data-based models of offshore wind turbines and bird migration.

Lee Spector is a Professor of Computer Science at Hampshire College and an Adjunct Professor of Computer Science at the University of Massachusetts, Amherst. He received a B.A. in Philosophy from Oberlin College in 1984 and a Ph.D. in Computer Science from the University of Maryland in 1992. He is the Editor-in-Chief of the journal Genetic Programming and Evolvable Machines and a member of the editorial board of Evolutionary Computation. He is also a member of the SIGEVO executive committee and he was named a Fellow of the International Society for Genetic and Evolutionary Computation.