Contents

Chapter 1

A FUNCTIONAL CROSSOVER OPERATOR FOR GENETIC PROGRAMMING

Josh Bongard[1]
[1]*Department of Computer Science, University of Vermont,* josh.bongard@uvm.edu

Abstract Practitioners of evolutionary algorithms in general, and of genetic programming in particular, have long sought to develop variation operators that automatically preserve and combine useful genetic substructure. This is often pursued with crossover operators that swap genetic material between genotypes that have survived the selection process. However in genetic programming, crossover often has a large phenotypic effect, thereby drastically reducing the probability of a beneficial crossover event. In this paper we introduce a new crossover operator, functional crossover (FXO), which swaps subtrees between parents based on the subtrees' functional rather than structural similarity. FXO is employed in a genetic programming system identification task, where it is shown that FXO often outperforms standard crossover on both simulated and physically-generated data sets.

Keywords: homologous crossover, crossover operators, system identification

1. Introduction

Genetic programming (Koza, 1992) refers to a family of algorithms that employ various data structures to represent candidate solutions to a given problem. These genotypes either produce behavior directly that is then selected, or are directly or indirectly transformed into a phenotype that in turn exhibits behavior which is subjugated to selection pressure. The choice of genetic encoding, the genotype to phenotype mapping, and the variation operators have a significant impact on the system's evolvability (Wagner and Altenberg, 1996), or ability to continually improve solutions.

The choice of variation operators is of particular interest in that they significantly affect how the population moves through the search space. Mutation operators are designed to discover better variants of a single genotype; crossover operators on the other hand should, when implemented properly, combine useful genetic substructure from multiple genotypes. Because most genetic programming instantiations are tree-based, crossover typically involves swapping subtrees between two parent trees, and this structural change often has a large phenotypic effect on the resulting genotypes. As originally articulated by Fischer (Fischer, 1930), the magnitude of the phenotypic effect of a genetic perturbation is inversely proportional to the probability of that perturbation being beneficial. For this reason it is often observed that random subtree crossover can adversely affect the performance of a genetic programming system. It may favor gradual increase in the size of genotypes over evolutionary time without providing any fitness benefit, a problem known as bloat (Langdon and Poli, 1997), and/or it may slow search by producing offspring that are less fit than their parents.

Several crossover operators have been proposed in the GP literature to improve their ability to combine useful genetic substructure from several parent genotypes. Headless chicken crossover (Jones, 1995) crosses subtreesbetween two GP trees in which one tree has survived selection while the second is created randomly in an attempt to introduce fresh genetic material into the population. Size fair crossover (Langdon, 1999) crosses subtrees between parent trees with a probability that is proportional to the size similarity between the selected subtrees. Homologous crossover refers to a family of crossover operators that attempt to preserve the context of the two crossed subtrees within their parent trees. D'haeseleer (D'haeseleer, 1994) has described deterministic and Langdon (Langdon, 1999) probabilistic homologous crossover operators that swap subtrees based on the similarity of their positions within their parent trees. Other homologous crossover operators based on syntactic similarity (Poli and Langdon, 1998; Nordin et al., 1999) have met with limited success.

Several researchers have argued that genetic material should be combined based on its semantic, rather than syntactic or structural similarity. Seman-

tic crossover (Beadle and Johnson, 2008) uses standard (random) crossover between two trees and then retains the resulting trees only if they differ semantically from their parents. In enzyme genetic programming (Lones and Tyrrell, 2001), genotypes are composed of independent elements that attach to one another based on their input and output characteristics. Crossover is accomplished by injecting elements from a donor into an existing genotype; the donated components will only be incorporated into the new genotype if they can connect to existing components.

In this paper we introduce a crossover operator that swaps subtrees based on their functional (semantic) rather than structural (syntactic) similarity, in an attempt to reduce the magnitude of the phenotypic effect of the cross. The next section describes this functional crossover (FXO) operator and its application to a system identification task. Section 3 contrasts FXO with standard crossover and no crossover, and section 4 provides some concluding remarks.

2. GP-based system identification

In previous work (Bongard and Lipson, 2007) genetic programming was applied to the problem of nonlinear system identification, in which coupled, nonlinear systems composed of multiple state variables are modeled as sets of ordinary differential equations. The system is composed of two components: a modeling and testing component. The modeling component uses genetic programming to evolve a population of models to describe a subset of time series data extracted from the system under study. The testing component uses the model population to derive a new set of initial conditions with which to perturb the system, and thereby generate new useful training data.

The algorithm proceeds as follows. Initially, a random set of initial conditions is provided to the target system, which generates a short tract of time series data in response. The modeling phase then commences by creating 15 random models and training them against this training data for 200 generations. A model's fitness is determined as its ability to reproduce as closely as possible the behavior of the target system when integrated starting with the same set of initial conditions.

Model evolution is then paused, and the testing component commences by creating 15 random sets of initial conditions. Each initial condition is provided to each of the current models, and the fitness of each set of initial conditions is determined as the rate of divergence in the models' predictions about how the system would respond to these initial conditions. The initial conditions are optimized for 200 generations, and the most fit set of initial conditions is provided to the target system, which generates a second tract of time series data in response. This second tract is added to the training set, and the modeling component recommences evolution with the current set of models, and

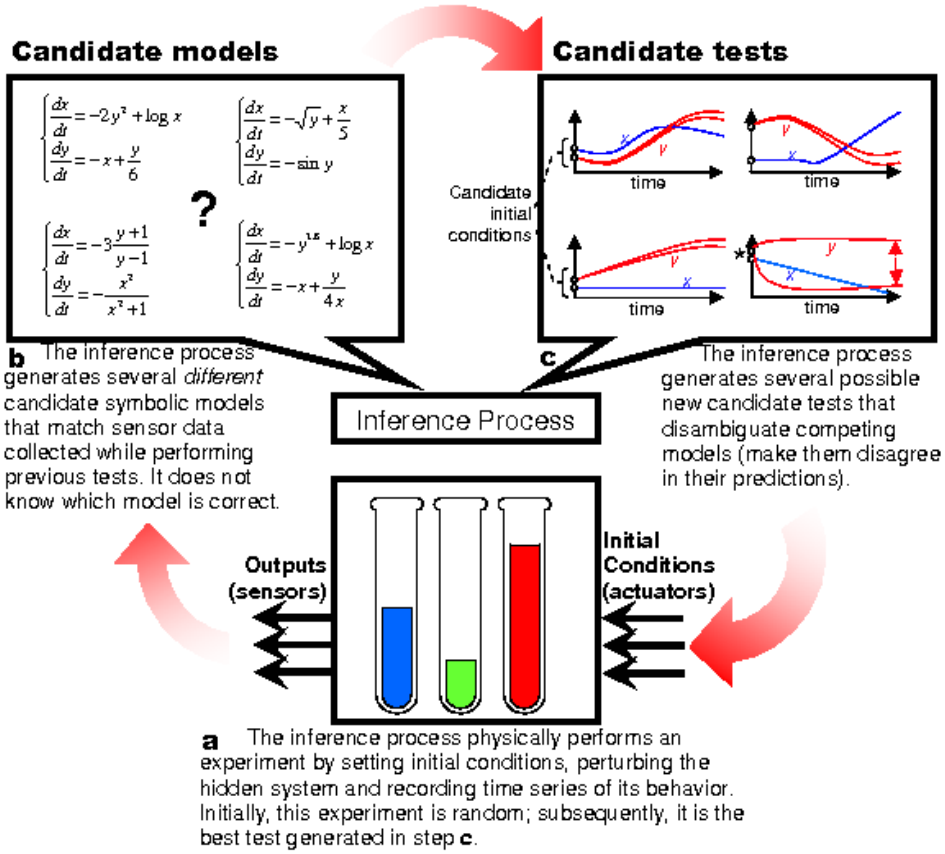

Figure 1-1. Overview of the GP-based system identification system.

re-optimizes them against both time series tracts in the training set for 200 generations. This cycle of system interrogation, modeling and testing is repeated a set number of times during each experiment, and is summarized in Fig. 1-1.

During modeling, it was found previously that integrating all of the ODEs describing each state variable together, and then computing the fitness of the model as a whole has low evolvability: If there is coupling between a well-modeled and a poorly-modeled state variable in a model, then that model will obtain an overall low fitness because the poorly-modeled ODE will drag the well-modeled state variable off course, and this well-modeled component will be lost during evolution. In (Bongard and Lipson, 2007) a technique called *partitioning* was introduced in which each ODE is integrated and evaluated separately, even though there may be coupling between the variables. This is accomplished as follows. During each time step of the integration of the ODE

describing a state variable, if there is a reference to another state variable in the GP tree then the value of that state variable generated by the target system, at that time step, is substituted into the terminal node. At the end of the modeling phase, the newly-optimized ODEs for each model are integrated back together to produce a full model. For more details about this methodology, please refer to (Bongard and Lipson, 2007).

Functional crossover

In the initial experiments using this system (Bongard and Lipson, 2007), crossover was not used as it was imperative to maintain variation in the population so that the testing component could induce disagreement amongst the predictions of the models for a given set of initial conditions, and it is well-known that crossover can reduce population hetereogeneity without necessarily conferring increased evolvability (e.g. (Bongard, 2007)). In order to improve the probability that crossover will incorporate useful genetic substructure into the receiving GP tree, a crossover operator that relies on semantic similarity between the two subtrees to be crossed was formulated and investigated here: functional crossover (FXO).

Given n state variables and 15 models, the population contains a total of $15n$ ODEs encoded as GP trees that are optimized. While each ODE is integrated, the minimum and maximum value that is passed upward by each node is recorded at that node. This process records the range of values experienced by each node during integration. After integration, the fitness of an ODE is computed as the error between the time series produced by the ODE and the time series produced by the target system for the corresponding state variable. A copy of each evaluated ODE is created, and the copy is mutated using standard GP mutation operators. The child ODE is integrated and evaluated: if its fitness is higher than its parent ODE, the parent is discarded and the child retained; otherwise, the child is discarded and the parent retained.

This experimental regime without crossover was contrasted to a second regime in which both mutation and standard GP crossover was employed. After all $15n$ ODEs are evaluated, they are copied and mutated. Within each of the n subgroups of 15 ODEs, a pair of the copies is chosen at random and crossed: a node is chosen at random in both trees, and the subtrees with those nodes as roots are swapped between trees. If either of the new trees is more fit than its parent, it is retained; otherwise, the new tree is discarded.

In the third regime, functional crossover is employed. Within each of the n subgroups of 15 ODEs, a pair of copies is chosen at random, and a node is chosen at random within the first tree. The node in the second tree is found that has the most similar range to that of the chosen node in the first tree, according

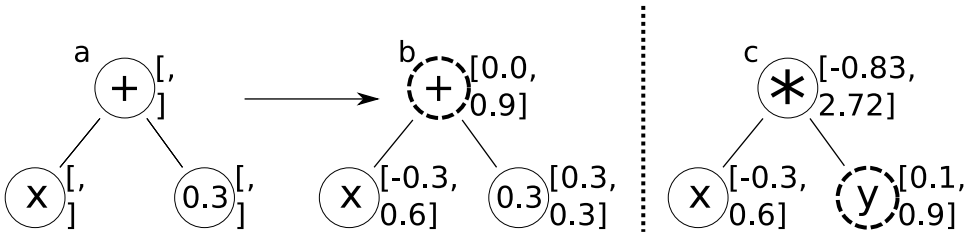

Figure 1-2. Functional crossover. While a GP tree is evaluated (a) the minimum and maximum values that pass through each node are recorded (b). If a node in the tree is then selected for crossover (b; dashed line), a second tree is chosen at random, and the node with the most similar range is found (c; dashed line), and those subtrees are then crossed as in standard GP crossover.

to

$$\min_{j=1}^{t}\left(\frac{|i_{\text{min}} - j_{\text{min}}| + |i\text{max} - j\text{max}|}{2}\right)$$

where t is the number of nodes in the second tree, i is the index of the node chosen from the first tree, and i_{min} and i_{max} are the minimum and maximum values passed upward by node i during integration, respectively. After finding the most similar node in the second tree, the two subtrees are crossed. In all other respects the third regime is identical to the first and second regimes. Functional crossover is illustrated in Fig 1-2.

3. Results

The three regimes were used to model both synthetic and physical systems. The first set of synthetic systems is shown in Table 3, and is composed of eighteen coupled, nonlinear systems with from 2 to 7 state variables. For each system, initial conditions for a state variable could range between zero and unity. Two hundred independent trials of the first regime, 200 trials of the second regime and 200 trials of the third regime were applied to each system. Each experiment was conducted for 40 cycles. At the end of each pass through the modeling component, the objective error of the best model was calcuated: the physical system generates time series that the model was not trained on, and the error of the model is calculated. The relative errors of the models produced by the three regimes is shown in Fig. 1-3, and the sizes of those models in Fig. 1-4.

For 6 of the 18 systems, functional crossover led to signficantly more accurate models than when either no crossover or standard crossover was employed (Fig. 1-3a,b,d,g,i,j). It can be noted that for these systems, FXO also tended to produce more compact models (Fig. 1-4a,b,d,g,i,j), despite the fact that there is no explicit selection pressure for smaller models. It is hypothesized that FXO produces more accurate and more compact models in these systems be-

	System 1	System 2	System 3
	a	b	c
$dx_1/dt =$	$-3x_1 - 3x_1x_2 + 2x_2x_2$	$-3x_1x_1 + 3x_1x_2 + 3x_2x_2$	$3x_1x_1 - x_1x_2 - x_2x_2$
$dx_2/dt =$	$-x_1x_1 - 3x_1x_2 - 2x_2x_2$	$-3x_1x_1 - 2x_1x_2 + 2x_2x_2$	$x_1x_1 + 3x_1x_2 - x_2x_2$
	d	e	f
$dx_1/dt =$	$-3x_1x3 - 2x_2x_3 - 3x_3x_3$	$-x_1x_2 + x_1x_3 - x_2x_3$	$-3x_1x_2 + x_1x_3 - x_3x_3$
$dx_2/dt =$	$-3x_1x_2 + x_1x_3 - 3x_2x_3$	$x_1x_1 + 2x_1x_2 + 2x_2x_3$	$-2x_1x_3 + 3x_2x_3 + 3x_3x_3$
$dx_3/dt =$	$3x_1x_2 + 3x_1x_3 - x_2x_3$	$-2x_1x_1 + x_1x_2 - 3x_2x_3$	$2x_1x_2 - 2x_1x_3 - 2x_2x_3$
	g	h	i
$dx_1/dt =$	$-x_1x_1 + 2x_2x_3 + 2x_3x_3$	$x_1x_4 + x_2x_4 + x_4x_4$	$-3x_1x_1 + 3x_1x_2 + 3x_2x_4$
$dx_2/dt =$	$x_1x_2 - 3x_1x_3 - 3x_2x_3$	$-3x_1x_2 - 2x_2x_3 - 3x_3x_4$	$-x_1x_1 - 2x_1x_3 - 3x_4x_4$
$dx_3/dt =$	$-x_1x_1 - x_2x_4 + 3x_4x_4$	$2x_1x_2 - x_1x_3 + 2x_2x_2$	$-2x_1x_4 + x_2x_2 - 3x_3x_4$
$dx_4/dt =$	$-3x_1x_2 - 3x_1x_4 - 3x_3x_4$	$x_1x_3 + 3x_2x_3 - x_3x_4$	$-x_1x_2 + 2x_1x_4 - 3x_3x_4$
	j	k	l
$dx_1/dt =$	$-3x_1x_5 + 3x_2x_3 - 3x_2x_5$	$-2x_2x_2 + 3x_3x_5 + 2x_4x_5$	$2x_1x_4 + 2x_2x_3 - x_2x_4$
$dx_2/dt =$	$-3x_1x_3 - 2x_3x_4 - x_4x_5$	$3x_1x_2 + x_1x_5 - 2x_2x_5$	$x_1x_3 + 3x_1x_4 + x_2x_4$
$dx_3/dt =$	$x_1x_1 - 3x_1x_4 + x_2x_4$	$x_1x_2 + 2x_2x_5 + 2x_4x_5$	$-2x_1x_1 + 2x_1x_2 - 3x_1x_3$
$dx_4/dt =$	$3x_1x_3 - 3x_1x_4 + 2x_2x_2$	$2x_1x_2 + 3x_1x_5 - x_4x_5$	$-3x_2x_5 + 3x_3x_4 - x_3x_5$
$dx_5/dt =$	$3x_1x_4 + 3x_3x_3 + 3x_3x_4$	$2x_1x_5 - x_2x_5 - 2x_5x_5$	$x_1x_1 + x_1x_5 + x_2x_3$
	m	n	o
$dx_1/dt =$	$-2x_1x_6 + x_2x_4 - 2x_2x_6$	$-2x_1x_3 - 3x_2x_4 + 2x_3x_6$	$x_1x_5 + x_1x_6 + x_4x_5$
$dx_2/dt =$	$x_1x_4 - x_1x_5 - 2x_4x_4$	$-3x_2x_4 + x_3x_4 - x_3x_6$	$-2x_2x_5 - 2x_2x_6 + 2x_3x_6$
$dx_3/dt =$	$2x_2x_5 - x_3x_4 + x_5x_5$	$-x_1x_2 - x_1x_3 + x_4x_6$	$-x_1x_5 - 2x_3x_4 + x_4x_4$
$dx_4/dt =$	$-3x_4x_5 - 2x_4x_6 + 2x_5x_5$	$-x_1x_4 + x_3x_5 - 2x_4x_6$	$3x_1x_2 + 3x_2x_3 - 2x_4x_5$
$dx_5/dt =$	$x_3x_6 - 2x_4x_4 - 3x_4x_5$	$3x_1x_2 - 3x_1x_6 - x_5x_5$	$-3x_1x_5 + x_2x_2 + 3x_2x_6$
$dx_6/dt =$	$x_3x_4 - x_3x_6 + 2x_4x_6$	$-3x_1x_3 - 2x_1x_6 - 3x_4x_6$	$-x_2x_5 - 2x_3x_5 - 3x_5x_6$
	p	q	r
$dx_1/dt =$	$-x_2x_2 - x_1x_2 + 3x_1x_1$	$-2x_2x_3 - 3x_1x_3 - 3x_3x_3$	$-3x_1x_1 + 3x_2x_2 + 3x_1x_2$
$dx_2/dt =$	$3x_1x_2 - x_2x_2 + x_1x_1$	$-3x_1x_2 - 3x_2x_3 + x_1x_3$	$-2x_1x_2 - 3x_1x_1 + 2x_2x_2$
$dx_3/dt =$	$2x_6x_7 - 2x_4x_4 + 3x_5x_7$	$3x_1x_3 - x_2x_3 + 3x_1x_2$	$-3x_4x_7 - 3x_3x_7 + 3x_4x_5$
$dx_4/dt =$	$x_3x_7 + 3x_3x_4 - 2x_4x_7$	$2x_5x_6 - x_4x_4 + 2x_6x_6$	$-2x_5x_6 - x_6x_7 - 3x_3x_5$
$dx_5/dt =$	$2x_4x_7 + 2x_6x_7 + x_3x_4$	$x_4x_5 - 3x_5x_6 - 3x_4x_6$	$x_4x_6 - 3x_3x_6 + x_3x_3$
$dx_6/dt =$	$3x_3x_7 - x_6x_7 + 2x_3x_4$	$3x_7x_7 - x_5x_7 - x_4x_4$	$2x_4x_4 + 3x_3x_5 - 3x_3x_6$
$dx_7/dt =$	$2x_3x_7 - 2x_7x_7 - x_4x_7$	$-3x_6x_7 - 3x_4x_5 - 3x_4x_7$	$3x_3x_6 + 3x_5x_5 + 3x_5x_6$

Table 1-1. The eighteen coupled nonlinear systems used for initial modeling.

cause FXO is able to swap out a large subtree that is an approximation of some function that can be expressed using fewer nodes, and therefore has a higher probability of swapping in a subtree from another tree that represents this function in a more compact way. For several of the other systems FXO produced more accurate models but not significantly so (Fig. 1-3e,h,n,o,p,q), and for no systems did the other two regimes signficantly outperform FXO.

The three regimes were also applied to four target systems that are manually-derived models of nonlinear mechanical (Pendulum), ecological (Lotka-Volterra) and biological (Lac operon) systems (Table 1-4). The initial values for each state variable in each system was restricted to the range $[0, 1]$. The models trained against the pendulum could be composed of algebraic and trigonometric functions; the Lotka-Volterra and high degree models were restricted to algebraic operators; and the Lac operon models were allowed algebraic functions and the Hill function ($x/(x+1)$). Terminal nodes were restricted to state variable references and floating-point constants. Fig. 1-5 reports the relative errors of the best models from 200 independent trials run using each of the three experimental regimes. Fig. 1-6 reports the relative sizes of these models.

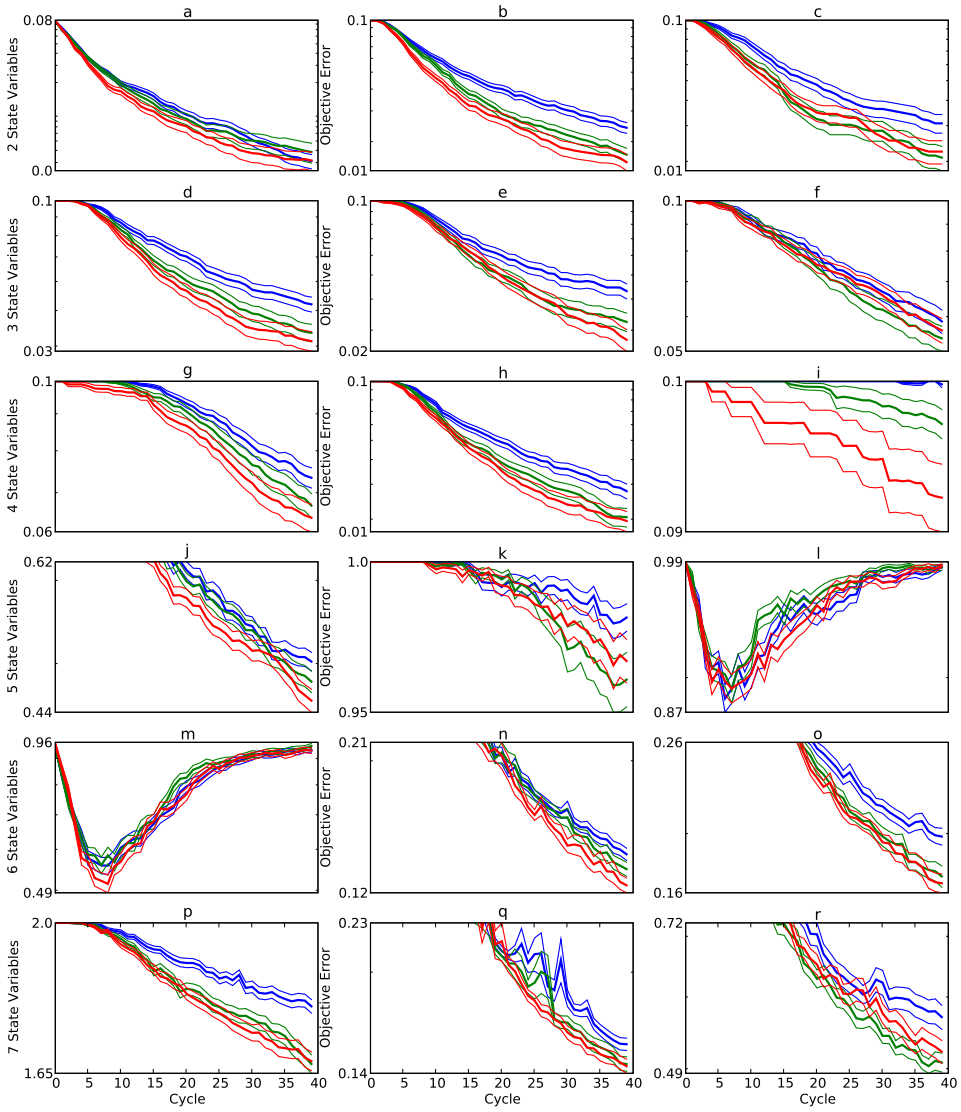

Figure 1-3. Relative modeling performance against the 18 synthetic systems using no crossover (blue), random crossover (green), and functional crossover (red). Thick lines indicate mean error; thin lines indicate one unit of standard error of the mean.

As can be seen in Fig. 1-5, functional crossover signficantly outperforms the other two regimes when employed for modeling the system of high degree and the Lac operon (Fig. 1-5c,d), provides some advantage for the Lotka-Volterra system (Fig. 1-5b), and provides a slight advantage for the pendulum, as does standard crossover. Fig. 1-6 indicates that for two of the systems functional

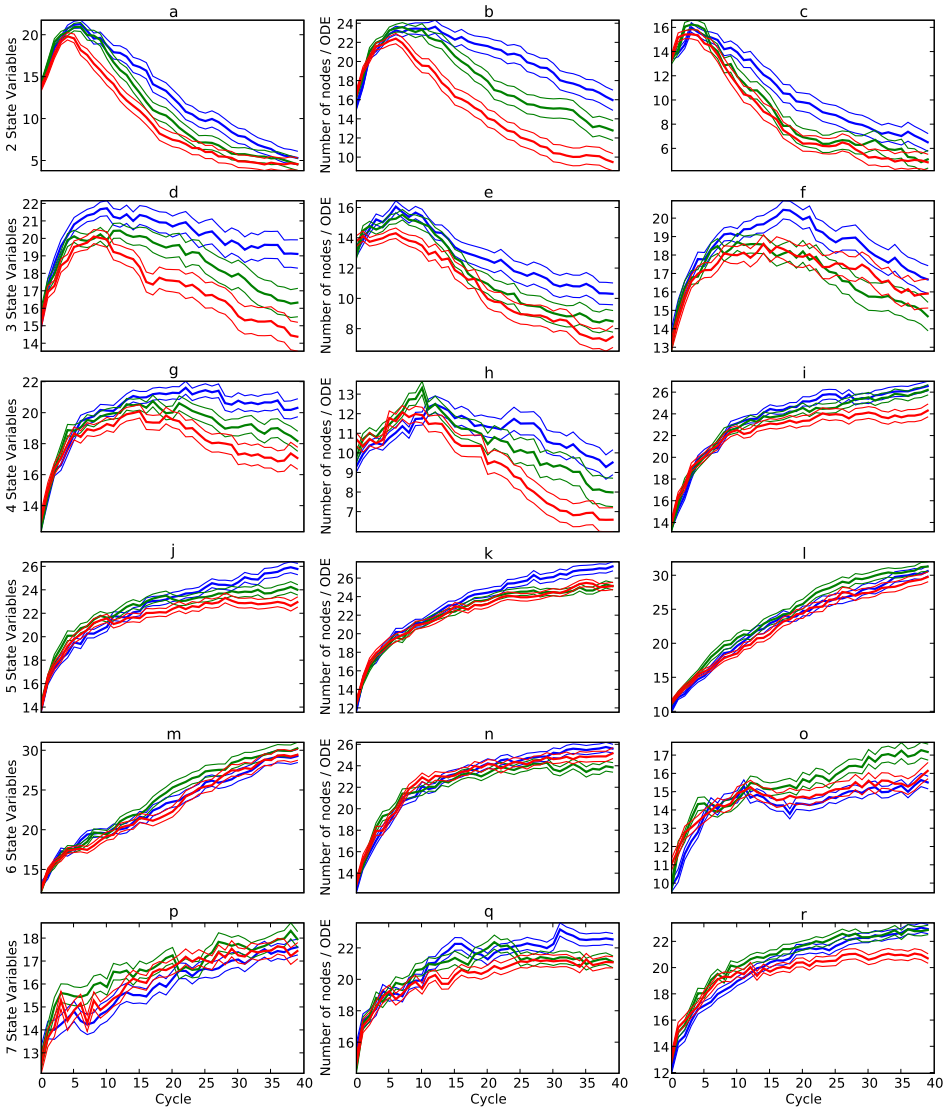

Figure 1-4. Relative model sizes resulting from the 18 synthetic systems using no crossover (blue), random crossover (green), and functional crossover (red).

crossover produces larger trees (Fig. 1-6a,c) while for the other two systems produces more compact trees (Fig. 1-6).

Finally, four physical systems were modeled by the three regimes. The first three systems are modifications of a physical pendulum, as shown in Fig. 1-7. Sensors in the pendulum's base record the angle of the arm relative to gravity,

a: Pendulum $d\theta/dt = \omega$ $d\omega/dt = -9.8\sin(\theta)$	b: Lotka-Volterra $dx/dt = 3x - 2xy - x^2$ $dy/dt = 2y - xy - y^2$
c: High Degree $dx/dt = -x^9y^{11}$ $dx/dt = -x^{11}y^9$	d: *Lac* operon $dG/dt = A^2/(A^2+1) - 0.01G + 0.001$ $dA/dt = G(L/(L+1) - A/(A+1))$ $dL/dt = -GL/(L+1)$

Table 1-2. The four synthetic target systems. a: A frictionless nondamped pendulum; b: two species competing for a common resource; c: a synthetic system with high degree; and d: a model of the Lac operon metabolic circuit in E. coli bacteria.

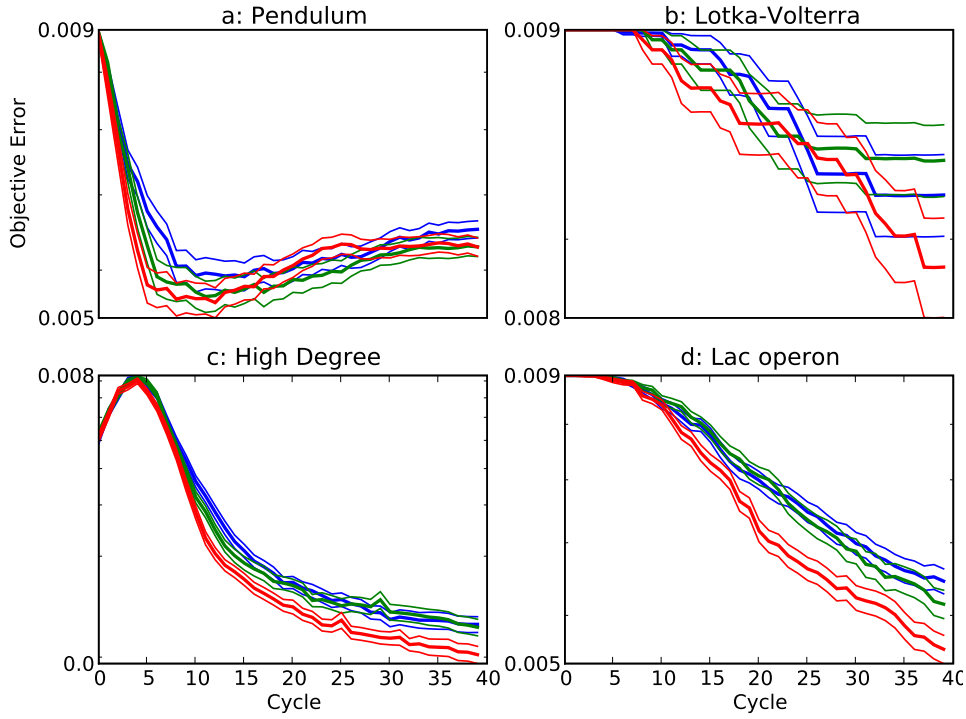

Figure 1-5. Relative modeling performance against the four target systems using no crossover (blue), random crossover (green), and functional crossover (red).

and angular velocity. The pendulum was swung and data was recorded when the base rested flat on a table (Fig. 1-7a); when the base was rotated 90 degrees counterclockwise (Fig. 1-7b), and rotated 138 degrees counterclockwise (Fig. 1-7c). The resulting data from these systems are reported as phase diagrams in Fig. 1-8a-c. The fourth physical system was a data set reflecting change

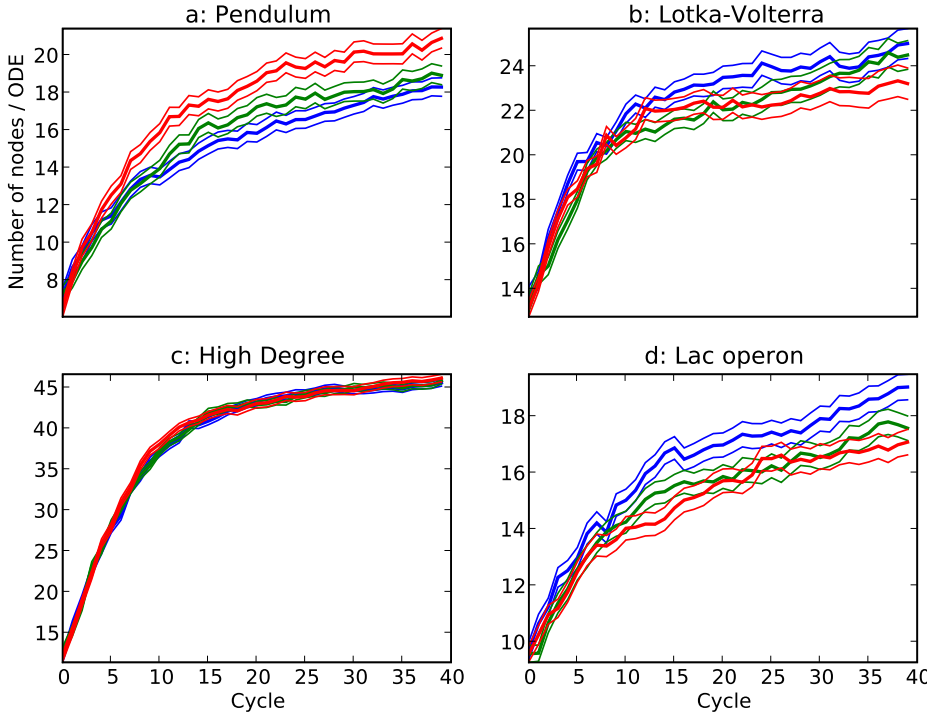

Figure 1-6. Relative model sizes resulting from the four target systems using no crossover (blue), random crossover (green), and functional crossover (red).

in population for the Canadian lynx and artic hare, as estimated by numbers of pelts recorded per winter by the Hudsons Bay Company (Odum and Odum, 1971). The data set indicates oscillations in both populations over a 100-year period (Fig. 1-8d).

Unlike the systems investigated so far, it is assumed that data has been previously generated by these systems, so the testing components cannot perturb the system based on model disagreement. Rather, the testing component searches for a time index within the existing data from the system for which, when the values for the state variables at that index are supplied to the models and the models are integrated, the models diverge in their predictions about future time indices. After a short period of optimization, the time index that induces maximum model disagreement, and the subsequent four time indices, are added to the training set.

For two of the data sets, crossover slows evolutionary search such that the regime with no crossover produces more accurate models (Fig. 1-9b,d). For the flat pendulum, there is no difference in model accuracy across the three experimental regimes. However for the pendulum when rotated 138 degrees

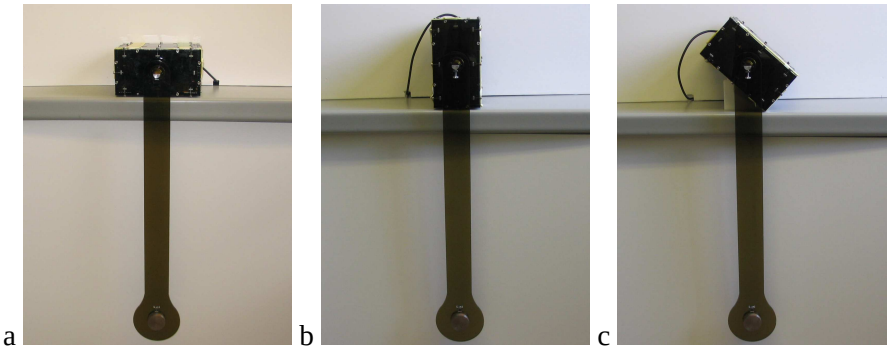

Figure 1-7. The pendulum was swung when it was in three different configurations, producing the data sets reported in Fig. 1-8a-c.

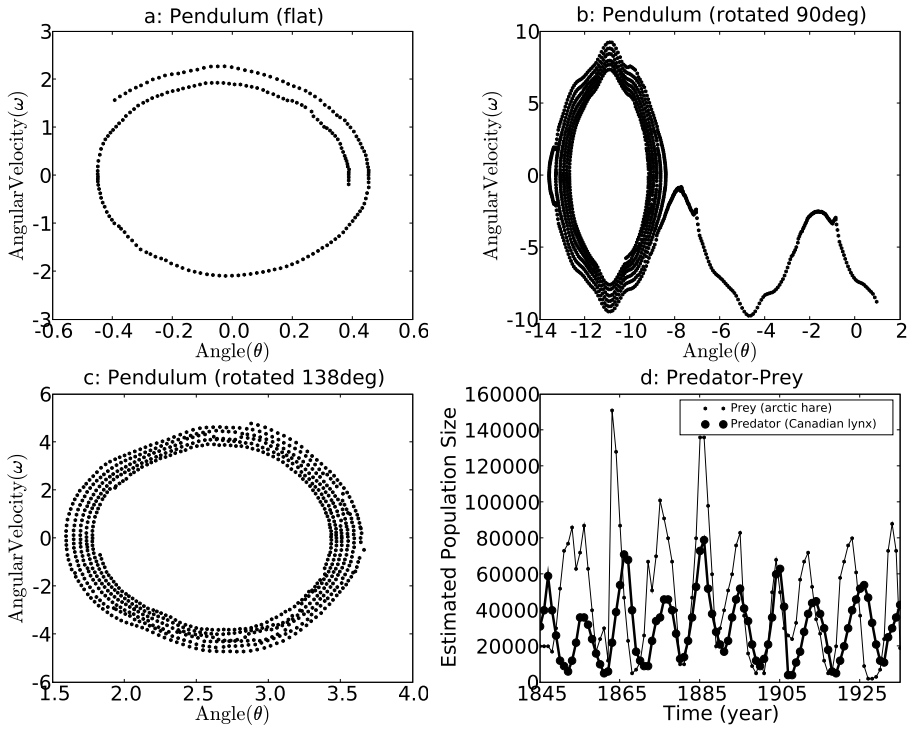

Figure 1-8. The four data sets produced by the four modeled physical systems. The data sets produced by the pendulum (a-c) are represented as phase diagrams; the ecological data set is drawn as a raw time series.

counterclockwise, functional crossover significantly outperforms the other two regimes. Unlike for the previous systems, functional crossover enlarges the size of models, compared to the other two experimental regimes.

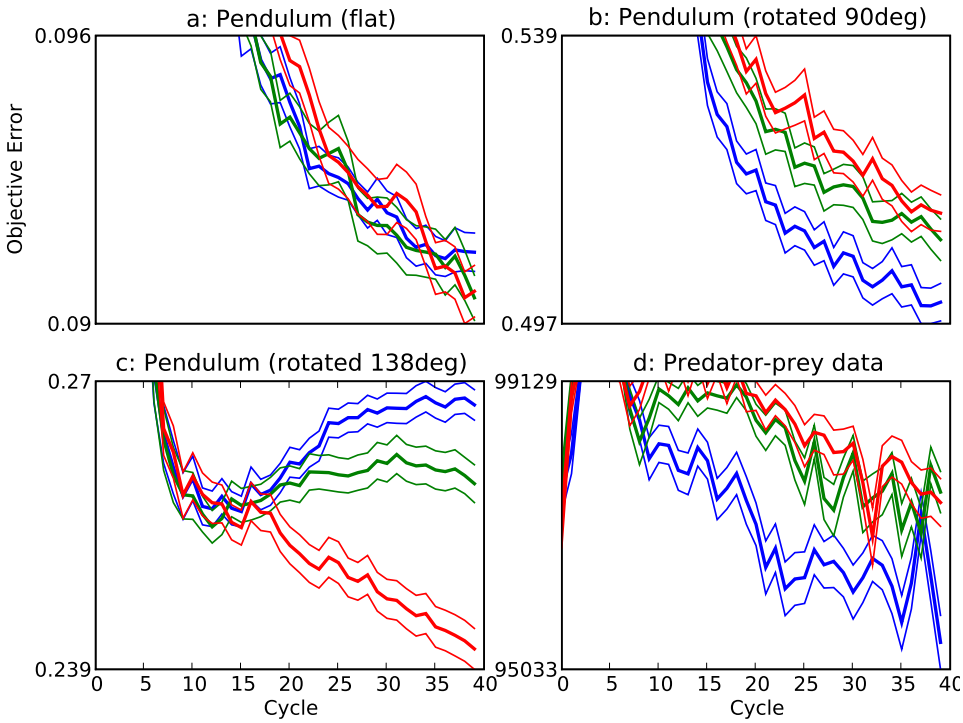

Figure 1-9. Relative modeling performance against the four physical systems using no crossover (blue), random crossover (green), and functional crossover (red).

4. Discussion and conclusions

In general, it was found that functional crossover helped to produce significantly more accurate models across a total of 26 synthetic and physical coupled, nonlinear systems. It is hypothesized that FXO confers an advantage because the phenotypic effect of a cross is less severe than a random crossover event. Because the two selected subtrees return values in a similar range, the newly-grafted subtree will return values similar to those returned by the original subtree, and will therefore impact the overall behavior of its parent tree less than if random crossover is employed. As has been known for some time (Fischer, 1930), a genetic perturbation has a higher probability of conferring an advantage the more mild the phenotypic effect of that perturbation is. It seems likely that this dynamic is the cause of the observed benefit of FXO, however more detailed investigation is required to validate this hypothesis.

It was found that for the first set of 18 systems, FXO tended to produce more accurate and more compact models. It is hypothesized that this is a result of FXO's ability to swap out a large subtree that approximates a function that can

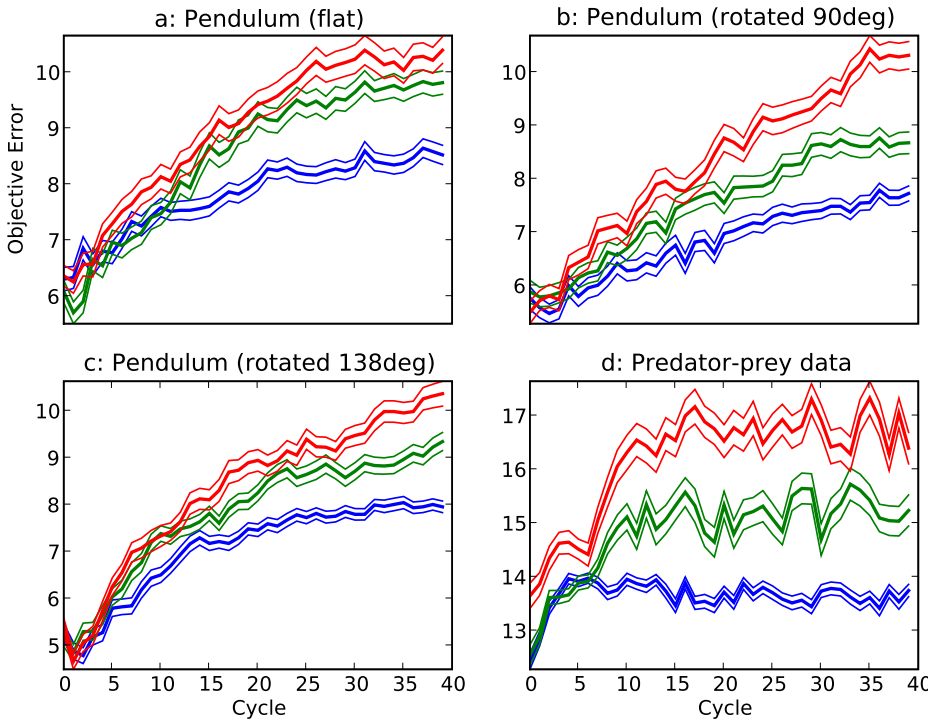

Figure 1-10. Relative model sizes resulting from the four physical systems using no crossover (blue), random crossover (green), and functional crossover (red).

be expressed with fewer nodes, and has a higher probability of swapping in a subtree from another tree that encodes this function more compactly, compared to random crossover. However, increased model accuracy was not accompanied by model compactness in the other systems. In particular for the physical systems, increased model accuracy (Fig. 1-9) was accompanied by an increase in model size (Fig. 1-10). It is believed that for the physical pendulum, an equation that would describe its friction may not be easily modelable, so the trees grow in size in an attempt to account for this.

Despite this, FXO was able to significantly outperform the no crossover and random crossover regimes for one of the pendulum arrangements (Fig. 1-9c). It seems likely the reason for this is that more data was collected for this arrangement compared to the data set associated with the flat pendulum (1-8a), and there is less noise present than in the data set collected from the pendulum rotated 90 degrees counterclockwise, which rotated over the top (1-8b).

Trees with similar structure may encode very different functions, which suggests that structural crossover operators for genetic programming may be of limited utility. Conversely, trees with very different structure may encode

similar functionality as neutral genetic structure tends to be prevalent in such systems, and the commutative properties of many operators admits to alternative encoding possibilities. The one current limitation of semantics-based crossover operators is that they are somewhat domain specific: tree structure tends to be similar across GP implementations, but the behavior of those trees is domain specific. That being said, FXO can still be employed in any GP system if trees are employed to compute a numerical function. It would also be relatively straightforward to develop semantic similarity metrics for other problem domains. For example, in the Central Place Food Foraging problem (Koza, 1992), a subtree describes a subset of an agent's behavior. During the evaluation of a tree, the area traversed by an agent as a result of each subtree may be recorded. FXO may then cross subtrees based on geometric similarities between the areas traversed by agents controlled by different trees.

Future work is planned in which this crossover operator will be compared directly to existing syntactic and semantic crossover opeators. Also, a probabilistic version of FXO (P-FXO) will be investigated in which subtrees are chosen from the second parent probabilistically, rather than deterministically based on nodes' functional similarity. Finally, rather than comparing minimum and maximum values experienced by nodes in different trees, probability distributions will be employed to better compare functional similarity between genetic substructure from different members of the population.

Acknowledgment

This work is supported in part by a 2007 Microsoft New Faculty Fellowship, and National Science Foundation grant EPS-0701410.

References

Beadle, Lawrence and Johnson, Colin (2008). Semantically driven crossover in genetic programming. In Wang, Jun, editor, *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 111–116, Hong Kong. IEEE Computational Intelligence Society, IEEE Press.

Bongard, J. (2007). Action-selection and crossover strategies for self-modeling machines. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 198–205. ACM New York, NY, USA.

Bongard, J. and Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943.

D'haeseleer, Patrik (1994). Context preserving crossover in genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, volume 1, pages 256–261, Orlando, Florida, USA. IEEE Press.

Fischer, R.A. (1930). *The Genetical Theory of Natural Selection*. Clarendon.

Jones, T. (1995). Crossover, macromutation, and population-based search. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 73–80. Morgan Kaufmann.

Koza, John R. (1992). A genetic approach to the truck backer upper problem and the inter-twined spiral problem. In *Proceedings of IJCNN International Joint Conference on Neural Networks*, volume IV, pages 310–318. IEEE Press.

Langdon, W. B. (1999). Size fair and homologous tree genetic programming crossovers. In Banzhaf, Wolfgang, Daida, Jason, Eiben, Agoston E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1092–1097, Orlando, Florida, USA. Morgan Kaufmann.

Langdon, W. B. and Poli, R. (1997). Fitness causes bloat. Technical Report CSRP-97-09, University of Birmingham, School of Computer Science, Birmingham, B15 2TT, UK.

Lones, Michael A. and Tyrrell, Andy M. (2001). Enzyme genetic programming. In *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*, pages 1183–1190, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea. IEEE Press.

Nordin, Peter, Banzhaf, Wolfgang, and Francone, Frank D. (1999). Efficient evolution of machine code for CISC architectures using instruction blocks and homologous crossover. In Spector, Lee, Langdon, William B., O'Reilly, Una-May, and Angeline, Peter J., editors, *Advances in Genetic Programming 3*, chapter 12, pages 275–299. MIT Press, Cambridge, MA, USA.

Odum, E.P. and Odum, H.T. (1971). *Fundamentals of Ecology*. Saunders Philadelphia.

Poli, Riccardo and Langdon, William B. (1998). Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation*, 6(3):231–252.

Wagner, G.P. and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976.